UNIVERSIDADE DO VALE DO TAQUARI – UNIVATES
CURSO DE ENGENHARIA DA COMPUTAÇÃO

VEHICLE-MANAGER: SISTEMA DE MONITORAMENTO VEICULAR

Alexandre Eger Marques

Lajeado, novembro de 2019

Alexandre Eger Marques

VEHICLE-MANAGER: SISTEMA DE MONITORAMENTO VEICULAR

Trabalho de conclusão de curso apresentado no Centro de Ciências Exatas e Tecnológicas, da Universidade do Vale do Taquari – Univates, como parte dos requisitos para obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Prof. Me. Alexandre Stürmer Wolf

Lajeado, novembro de 2019

AGRADECIMENTOS

Agradeço em primeiro lugar os meus pais, em especial a minha mãe que lutou para que esse momento acontece-se, agradeço o meu orientador Alexandre Wolf por toda ajuda e auxílio prestado para concepção deste trabalho.

Agradeço a empresa Interact Solutions por ter me dado a oportunidade, espaço e recursos para realizar o desenvolvimento deste projeto.

Agradeço a Deus pela força e coragem que permitiram atingir meu objetivo e realizar o meu sonho. Muito obrigado.

RESUMO

Com a evolução da eletrônica e o desenvolvimento de dispositivos embarcados, os automóveis passaram a abranger tecnologias como o sistema de diagnóstico OBDII. Ao passo que os sistemas embarcados dos veículos melhoravam, os dispositivos de controle eram alocados para uma unidade central e comunicavam-se por uma rede local, possibilitando a transmissão de dados de todos os módulos do automóvel e facilitando a aquisição de dados sobre o estado do veículo. O presente trabalho objetiva a elaboração de uma solução de monitoramento para veículos, onde as informações obtidas alimentarão um sistema web. Tais informações são enviadas por meio de um aplicativo Android, o qual realiza a comunicação entre o sistema de diagnóstico e o servidor de banco de dados. A utilização do aplicativo Android justifica-se por dispensar a criação de um hardware, reduzindo assim o custo da solução. Para atingir os objetivos propostos, é realizada a pesquisa e a análise das tecnologias de diagnóstico utilizadas em veículos automotores, testes de bancada e testes em campo, além do estudo da construção da solução de monitoramento via aplicativo Android e sistema web. Ao final do trabalho, conclui-se que a solução apresenta grande potencial, possibilitando um monitoramento moderno do estado do veículo.

Palavras-chave: OBDII. Android. Sistemas embarcados. Monitoramento veicular. ELM327.

ABSTRACT

With the evolution of electronics and the development of embedded devices, automobiles have come to encompass technologies such as the OBDII diagnostic system. As on-board vehicle systems improved, control devices were allocated to a central unit and communicated over a local area network, enabling data transmission from all car modules and facilitating the acquisition of vehicle status data. vehicle. The present work aims the elaboration of a monitoring solution for vehicles, where the obtained information will feed a web system. This information is sent through an Android application, which communicates between the diagnostic system and the database server. Using the Android application is justified by the need to create hardware, thus reducing the cost of the solution. To achieve the proposed objectives, the research and analysis of diagnostic technologies used in automotive vehicles, bench tests and field tests, as well as the study of the construction of the monitoring solution via Android application and web system is performed. At the end of the work, it is concluded that the solution has great potential, enabling a modern monitoring of the state of the vehicle.

Keywords: OBDII. Android. Embedded systems. Vehicle monitoring. ELM327.

LISTA DE FIGURAS

Figura 1 – Módulo do sistema automotivo e seus componentes.....18

Figura 2 – ECU.....22

Figura 3 – Conector OBDII 16 pinos 23

Figura 4 – Rede CAN veicular.....25

Figura 5 – *Scanner* de leitura ELM327.....27

Figura 6 – *Workflow* para desenvolvimento de *softwares* 29

Figura 7 – Arquitetura do sistema operacional Android.....31

Figura 8 – Clientes e servidor se comunicando por meio da *Internet*.....32

Figura 9 – Sete camadas do modelo OSI.....33

Figura 10 – Rede de comunicação *Bluetooth* entre dispositivos.....34

Figura 11 – Arquitetura MVC.....35

Figura 12 – Arquitetura de monitoramento de veículos.....41

Figura 13 – Protótipo da aplicação *mobile* desenvolvida para testes.....43

Figura 14 – Entrada OBDII do veículo.....44

Figura 15 – *Scanner* ELM327 plugado na porta OBDII do veículo.....45

Figura 16 – Protótipo de aplicação *mobile*, com dados da ECU em tempo real.....46

Figura 17 – Esquema de integração entre os sistemas.....47

Figura 18 – Método Java para leitura de dados e envio para o servidor 48
Figura 19 – Entidade com modelo do cadastro de veículos 51
Figura 20 – Diagrama ER-1 com modelo de registros de viagens 52
Figura 21 – Diagrama ER-2 com modelo de informações da ECU e geolocalização53
Figura 22 – Editor de registro de veículos 55
Figura 23 – Veículo cadastrado no sistema 55
Figura 24 – Tela de *login* do aplicativo *mobile* 56
Figura 25 – Tela do seletor de veículos.....57
Figura 26 – Tela do seletor de veículos sem registros 58
Figura 27 – Tela do registro de viagens 59
Figura 28 – Tela de processamento e envio de dados para o servidor 60
Figura 29 – Tela de finalização de viagem e interrupção do envio de dados 61
Figura 30 – Condução do veículo e envio de dados 62
Figura 31 – Tabela de listagem das viagens em andamento e realizadas 63
Figura 32 – Informações detalhadas da viagem.....63
Figura 33 – Relatório de informações da ECU do veículo 64
Figura 34 – Geolocalização do veículo.....65

LISTA DE EQUAÇÕES

Equação 1 – Equação para cálculo de rotações do motor 26

LISTA DE ABREVIATURAS E SIGLAS

ACU	Airbag Control Unit – Unidade de Controle do Airbag
CAN	Controller Area Network
CCU	Convenience Control Unit
CRC	Cyclic Redundancy Check – Verificação de Redundância Cíclica
CSS	Cascading Style Sheets – Folha de Estilo em Cascatas
DTC	Diagnostic Trouble Code
DTM	Diagnostic Test Mode
ECM	Engine Control Module – Módulo de Controle do Motor
ECU	Engine Control Unit – Unidade de Controle do Motor
FHSS	Frequency-Hopping Spread Spectrum – Espectro de Difusão em Frequência Variável
HTML	Hypertext Markup Language – Linguagem de Marcação de Hipertexto
HTTP	Hypertext Transfer Protocol – Protocolo de Transferência de Hipertexto
IEEE	Institute of Electrical and Electronics Engineers – Instituto de Engenheiros Eletricistas e Eletrônicos
ISO	International Organization of Standardization – Organização Internacional de Normalização
JRE	Java Runtime Environment – Ambiente de Execução Java
JSON	JavaScript Object Notation
JVM	Java Virtual Machine – Máquina Virtual Java
MAC	Media Access Control Address
OBD	On-Board Diagnostic

OSI	Open System Interconnection
PDU	Protocol Data Unit – Unidade de Dados de Protocolo
PID	Parameter ID
PWM	Pulse Width Modulation – Modulação de Largura de Pulso
REST	Representational State Transfer
RIA	Rich Internet Application – Aplicação de Internet Rica
RPM	Rotações por Minuto
SAE	Society of Automotive Engineers – Sociedade de Engenheiros de Automóveis
SDK	Software Development Kit – Kit de Desenvolvimento de Software
TCP/IP	Transmission Control Protocol/Internet Protocol – Protocolo de Controle de Transmissão/Protocolo de Internet
TCU	Transmission Control Unit – Unidade de Controle de Transmissão
VPW	Variable Pulse Width – Largura de Pulso Variável
W3C	World Wide Web Consortium
Wi-Fi	Wireless Fidelity
XML	Extensible Markup Language – Linguagem de marcação extensível

SUMÁRIO

1 INTRODUÇÃO 12
1.1 Objetivos 15
1.1.1 Objetivo geral 15
1.1.2 Objetivos específicos 15
1.2 Estrutura do trabalho 15

2 REFERENCIAL TEÓRICO 17
2.1 Eletrônica automotiva embarcada 17
2.1.1 Sistemas embarcados 18
2.1.2 Sensores 19
2.1.3 Atuadores 20
2.1.4 Engine Control Unit (ECU) 20
2.1.5 On-Board Diagnostic II (OBDII) 22
2.2 Protocolos de comunicação veicular 23
2.2.1 Protocolo SAE J1850 Pulse Width Modulation (PWM) 24
2.2.2 Protocolo SAE J1850 Variable Pulse Width (VPW) 24
2.2.3 Controller Area Network (CAN) 24
2.3 Tecnologias de diagnóstico 25
2.3.1 ELM327 26
2.4 Tecnologia da informação 27
2.4.1 Software 28
2.4.2 Desenvolvimento de software 28
2.4.3 Aplicativos mobile 29
2.4.4 Android 30
2.4.5 Servidores Web 31
2.5 Comunicação de dados 32
2.5.1 TCP/IP 33
2.5.2 Bluetooth 34
2.6 Aplicação web 35

2.6.1 HTML e CSS 36
2.6.2 ZK Framework 36
2.6.3 Java 37

3 METODOLOGIA 39
3.1 Métodos de pesquisa 39
3.2 Cenário 40
3.3 Aplicações 40
3.4 Procedimentos técnicos usados na pesquisa 41

4 DESENVOLVIMENTO 42
4.1 Desenvolvimento do protótipo de validação 42
4.1.1 Interface de comunicação veicular 43
4.1.2 Implementação do scanner ELM327 no veículo 45
4.1.3 Validação do protótipo 46
4.2 Integração entre os sistemas 47
4.3 Desenvolvimento da aplicação mobile 48
4.4 Desenvolvimento do webservice 49
4.5 Desenvolvimento do sistema web 49
4.6 Requisitos e funcionalidades 50
4.6.1 Registro de veículos 51
4.6.2 Registro de viagens 52
4.6.3 Envio de informações do veículo e da geolocalização 52

5 RESULTADOS 54
5.1 Cadastro de veículo 54
5.2 Autenticação no aplicativo mobile 56
5.3 Seletor de veículo cadastrado 57
5.4 Registro de viagens 58
5.5 Envio de dados para o servidor 59
5.6 Finalização do processo de envio de dados 60
5.7 Relatório de informações da viagem 61
5.7.1 Envio de informações do veículo e da geolocalização 64
5.8 Validação 65
5.9 Dificuldades 66

6 CONCLUSÃO 68

REFERÊNCIAS BIBLIOGRÁFICAS 70

1 INTRODUÇÃO

Com a evolução tecnológica da eletrônica, alguns sistemas mecânicos passaram a ser substituídos por dispositivos eletrônicos que realizam a mesma função, de forma mais eficiente e com baixo consumo energético. Essas tecnologias foram empregadas em sistemas computacionais, industriais, automotivos, entre muitos outros. E, a partir desse desenvolvimento tecnológico, os veículos começaram a integrar sistemas eletrônicos sofisticados em seu interior, visando o controle, o diagnóstico, a manutenção, o consumo, a redução de custos e a segurança.

Segundo Capelli (2010), os sistemas eletrônicos embarcados em veículos possuem uma taxa de crescimento exponencial desde a década de 1990, onde a tendência era de aperfeiçoar o sistema para obter um rendimento melhor do veículo. Essa ideia de melhorias do sistema teve início na década de 1920, com o surgimento das primeiras injeções eletrônicas que evoluíram a partir dos mecanismos de injeção direta de combustível. Segundo o autor, essa tecnologia foi aprimorada durante o ano de 1936 por Daimler-Benz e utilizada em aviões no período da Segunda Guerra Mundial. O autor ainda afirma que toda tecnologia de aeronaves acaba chegando para os veículos em algum momento.

O sistema de injeção eletrônica melhorou para sempre a performance e o rendimento do veículo. Capelli (2010) afirma que esse sistema foi implementado em um veículo pela primeira vez no ano de 1954, onde o modelo era um Mercedes Benz 300 SL. Com isso, começaram a surgir novos conceitos para o gerenciamento veicular e posteriormente são desenvolvidas as primeiras centrais de gerência e

controle para motores, a *Eletronic Control Unit* (ECU) e a *Engine Control Module* (ECM).

O princípio do desenvolvimento dessas centrais era diminuir o consumo, mas dois fatores foram decisivos para o surgimento desse sistema: a diminuição da poluição provocada pelos motores à explosão e a crise do petróleo de 1973 (CAPELLI, 2010).

Esse módulo de gerenciamento tem como principal objetivo o controle da injeção de combustível, o controle da mistura, o controle da ignição e da memória adaptativa. Conforme Capelli (2010), a ECU é uma caixa metálica com conexões para os sensores e atuadores do veículo. Os sensores são responsáveis por enviar as informações obtidas do estado dos componentes do veículo para a ECU, e a partir desses dados enviados a mesma exerce determinados controles no motor, podendo variar conforme o modelo.

Conforme Guimarães (2007), durante a década de 1970, surgiu a instrumentação eletrônica dos sistemas automotivos, embutindo o sistema de indicadores de alertas, sistemas de gerenciamento eletrônico em diversas partes do veículo, exibição de dados relacionados ao diagnóstico dos diversos sistemas eletrônicos veiculares e aquisição de dados de condução. O conjunto dessas funções fazem parte da diagnose *on-board*, localizada no painel do veículo.

Conforme o desenvolvimento dos sistemas de gerenciamento veiculares e suas redes de comunicação entre os módulos evoluíam, surgiu a necessidade de realizar esses diagnósticos em campo, seja durante a fabricação ou manutenção dos veículos. Alguns fabricantes desenvolveram *scanners* para comunicar-se com a rede interna do veículo por meio de uma comunicação serial e a partir desta realizar uma leitura dos módulos de gerenciamento veicular.

De acordo com Guimarães (2007), ao longo dos anos o processo de diagnóstico veicular passou por diversas etapas de padronização, até que em 1996 foi implementado em todos os veículos americanos o sistema *On-Board Diagnostic* (OBD), a primeira versão, onde o mesmo realizava uma verificação dos códigos de erros armazenados nos módulos veiculares e disponibilizando os resultados através

de uma porta de diagnóstico. Esse sistema de diagnóstico receberia uma série de atualizações, até o lançamento do *On-Board Diagnostic* II (OBDII).

O objeto deste estudo é uma frota de veículos de uma empresa localizada no Vale do Taquari. Atualmente os veículos dessa empresa não possuem nenhum gerenciamento sobre o estado do veículo, apenas anotações de quilometragens de saída e chegada em uma planilha física, além de não existir nenhuma forma de monitorar as rotas e a geolocalização do veículo. Desta forma, o gerenciamento desses automóveis é muito limitado, restringido apenas a informações básicas reportadas pelo condutor em uma planilha física.

A proposta desse trabalho consiste no desenvolvimento de uma solução para o gerenciamento sobre o estado dos veículos dessa empresa, visto a necessidade de possuir um controle mais eficiente e um gerenciamento mais moderno para melhorar a durabilidade, os gastos da frota e sua manutenção. Esse sistema abrange o controle de quilometragem, temperatura do motor, rotas realizadas pelo veículo, percentual de combustível, aceleração, velocidade média, rotações do motor e pressão barométrica, esses valores são obtidos de uma unidade central do veículo e enviados para um servidor de banco de dados, além de salvar as rotas realizadas pelo veículo por meio do envio das coordenadas do gps do *smartphone*.

O desenvolvimento dessa solução foi dividido em dois grandes processos. No primeiro momento, ocorreu o desenvolvimento de um aplicativo *mobile* para plataforma Android, denominado de conector utilizando a linguagem Java, e o mesmo permite a comunicação do sistema de diagnóstico com uma base de dados. Já a segunda parte compreendeu o desenvolvimento de um sistema de gerenciamento dos veículos em ambiente *web*, utilizando a linguagem Java. O andamento desse processo deu-se nessa ordem, pois sem o conector de informações o envio de dados do veículo para o banco de dados não seria possível.

1.1 Objetivos

1.1.1 Objetivo geral

Gerenciar o estado dos veículos de uma frota e a sua geolocalização por meio do envio de informações coletadas da central do automóvel e as coordenadas do gps do *smartphone*, apresentado esses valores obtidos em um sistema *web*.

1.1.2 Objetivos específicos

- Efetuar um estudo sobre as tecnologias de diagnóstico presente em veículos;
- Definição das tecnologias necessárias para o desenvolvimento do projeto;
- Disponibilizar um referencial teórico e prático para que sejam desenvolvidos futuros trabalhos a partir desta pesquisa;
- Desenvolvimento de um aplicativo *mobile* para comunicar-se com a central do veículo e com o servidor de banco de dados;
- Desenvolvimento de um sistema *web* para exibir as informações enviadas do veículo aos usuários do sistema, como dados dos sensores e geolocalização;
- Realizar um experimento prático, onde serão efetuadas conduções do veículo para obter as informações necessárias e alimentar os módulos do sistema *web*.

1.2 Estrutura do trabalho

A estruturação deste trabalho está dividida em 6 capítulos. O primeiro capítulo possui a introdução sobre as tecnologias de diagnóstico veicular e suas evoluções. No segundo capítulo é retratado o referencial teórico, que auxiliou na construção do trabalho ao definir as tecnologias de programação utilizadas, os protocolos de

comunicação e os demais itens para realização da proposta apresentada. Os principais tópicos trabalhados são a eletrônica embarcada veicular, as tecnologias de diagnóstico, tecnologia da informação, comunicação de dados, aplicação *web* e aplicação *mobile*. No terceiro capítulo é abordado a metodologia no qual o trabalho foi embasado, além de retratar sobre o cenário que será aplicado a solução e a proposta apresentada para o desenvolvimento. No quarto capítulo é apresentado o desenvolvimento da aplicação *mobile*, *webservice*, sistema *web* e implantação em ambiente de produção. No quinto capítulo é apresentado o funcionamento das aplicações *web* e *mobile*, além de apresentar as validações e as dificuldades encontradas no desenvolvimento deste trabalho. O sexto capítulo apresenta a conclusão deste trabalho, encerrando todas as considerações. Por fim, as referências bibliográficas são apresentadas, aos quais embasaram o desenvolvimento deste trabalho.

2 REFERENCIAL TEÓRICO

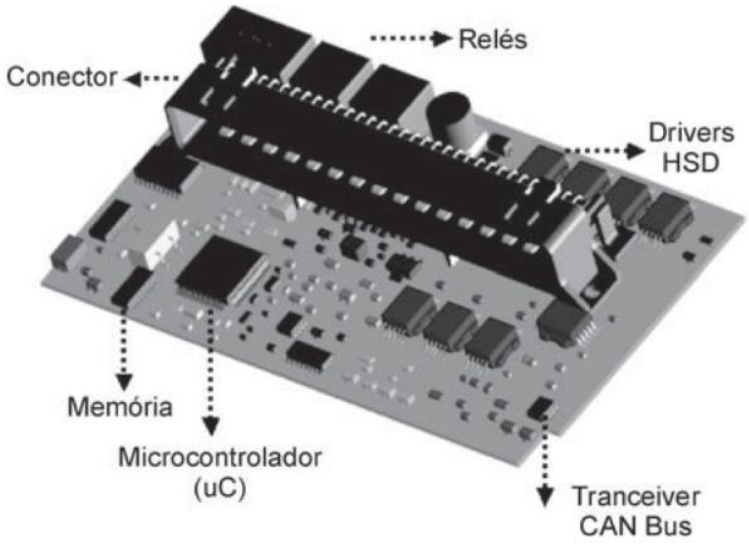
Para o desenvolvimento deste trabalho, foram estudados os sistemas embarcados dos veículos, eletrônica veicular e seu sistema de diagnóstico OBDII, além de sua rede de comunicação e protocolos, onde os mesmos serão apresentados nesta seção.

2.1 Eletrônica automotiva embarcada

Segundo Guimarães (2007), seja qual for o sistema eletrônico embarcado deve ser alimentado por uma fonte de energia, apresentar entradas e saídas para conexão de sensores e atuadores, e esses sistemas necessitam de algum tipo de proteção contra surtos e curtos-circuitos, como fusíveis que são instalados para proteger o sistema eletrônico. O sistema eletrônico de um veículo contém relés, chicotes para conexão entre os dispositivos eletrônicos, fusíveis para proteção do sistema e sensores. Esses componentes são essenciais para formação dos sistemas de eletrônica embarcada veicular.

A Figura 1 apresenta uma placa do sistema eletrônico embarcado de um veículo, elaborada com base na obra de Guimarães (2007).

Figura 1 – Módulo do sistema automotivo e seus componentes



Fonte: Adaptado pelo autor com base em Guimarães (2007).

Para um entendimento melhor sobre o sistema eletrônico embarcado de um veículo, é necessário conceituar e apresentar os seus componentes de forma individual, e eles serão apresentados a seguir.

2.1.1 Sistemas embarcados

Segundo Oliveira e Andrade (2010), a composição de um sistema embarcado é constituída de uma unidade de processamento fixada em um circuito impresso. Esses sistemas embarcados possuem a capacidade de processar informações realizadas por um *software* denominado de *firmware*, que está sendo processado no interior dessa unidade. Conforme o autor aborda em seu livro, um sistema embarcado é formado de memórias, microcontrolador ou microprocessador, barramentos e sistemas de comunicação. Esses sistemas estão embutidos em eletrodomésticos, dispositivos industriais, automóveis etc. O objetivo principal dos sistemas embarcados é atender um sistema muito maior.

Os sistemas embarcados são fundamentais no setor automobilístico. Conforme Capelli (2010), um automóvel possui diversos sistemas embarcados encarregados de efetuar determinadas tarefas no veículo, como os sistemas de controle elétrico, alarmes e imobilizadores, painel de instrumentos, *airbags*, freios e

a rede *Controller Area Network* (CAN), responsável por conectar todos os dispositivos eletrônicos de um veículo.

2.1.2 Sensores

Segundo Thomazini e Albuquerque (2011), com a necessidade de estipular as condições de determinado sistema, é essencial obter os valores das variáveis físicas de determinado local que está sob monitoramento, sendo assim a definição de sensores, que são dispositivos responsáveis por verificar e interferir em ambientes que estão implementados.

Os sensores são dispositivos sensíveis a determinadas formas de energia localizadas em algum ambiente, podendo variar entre térmica e luminosa, realizando um relacionamento com as informações de grandeza, como temperatura, aceleração, velocidade, entre outras (THOMAZINI E ALBUQUERQUE, 2011).

Os sensores podem ser divididos em analógicos e digitais. Os sensores analógicos podem assumir qualquer valor em um período, como pressão, temperatura, velocidade, vazão, torque, força e luminosidade. Já os sensores digitais só assumem valores binários em suas saídas (0 ou 1), sendo utilizado na detecção de objetos. Os sensores possuem diversas características importantes, como o seu tipo de saída que varia entre valores analógicos ou digitais, sua sensibilidade que objetiva fornecer uma variação em sua saída para determinada grandeza, a exatidão que retorna um valor obtido como verdadeiro e o valor encontrado na análise, e por fim a sua precisão em gerar uma concordância de um valor em um determinado período, sob as mesmas condições e com o mesmo equipamento (THOMAZINI E ALBUQUERQUE, 2011).

Conforme Laganá (2019), existem diversos sensores de monitoramento em um veículo, como sensores de temperatura para monitorar o motor e demais locais, sensores de pressão para monitoramento dos pneus, sensores acelerômetros para medir vibrações e velocidade, sensores de rotação para calcular a velocidade de rotação do eixo e sensores de relutância que servem para realizar medidas sobre a velocidade e posição de componentes metálicos em movimento. Todos esses

dispositivos instalados em um automóvel contribuem para uma melhor segurança, rendimento do veículo e detecção de problemas.

De acordo com Saloman (2012), existe uma grande importância dos sensores para o século XXI, pois são utilizados em indústrias, automóveis e diversos equipamentos. Esses dispositivos realizam o sensoriamento de inúmeros objetos, prevenindo problemas, desastres e auxiliando em uma melhor produção, eficiência e segurança. Conforme o autor aborda em seu livro, a gama de sensores presentes no dia a dia é imensa e isso contribui para o desenvolvimento tecnológico.

2.1.3 Atuadores

Segundo Thomazini e Albuquerque (2011), os atuadores são dispositivos capazes de controlarem e modificarem variáveis, pois recebem determinado sinal de um controlador e atuam sobre o sistema controlado, resultando em alguma modificação física. Os atuadores podem ser classificados como motores, válvulas, relés, cilindros etc.

2.1.4 Engine Control Unit (ECU)

Para Capelli (2010), o gerenciamento eletrônico de um veículo foi testado pela primeira vez em 1939. Esses testes foram realizados pela Bosch em aeronaves, onde ocorreu a primeira implementação do sistema de injeção direta de gasolina. Já na década de 1950 foi produzido o primeiro veículo com injeção eletrônica. Durante esse período, o gerenciamento eletrônico veicular possuía a capacidade de apenas controlar a injeção de combustível.

As centrais atuais realizam muito mais funções e não gerenciam apenas o sistema de injeção. Os veículos mais modernos possuem diversos sistemas de controle, cada qual gerenciando o seu sistema, como:

- *Engine Control Unit* (ECU), responsável pelo gerenciamento do motor;

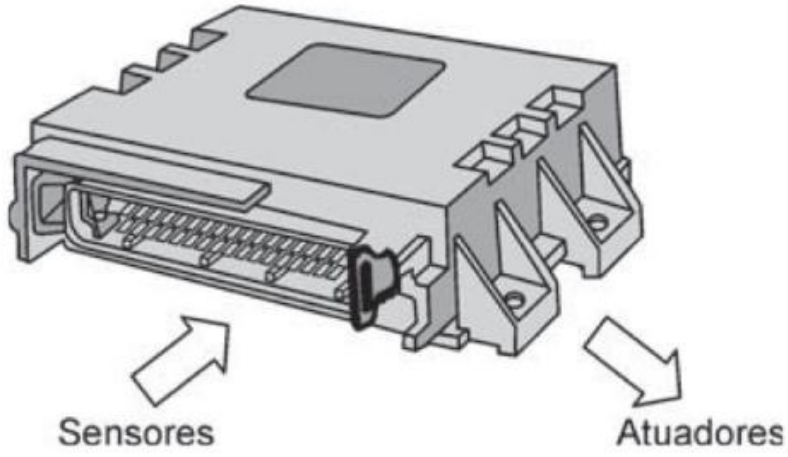
- *Airbag Control Unit* (ACU), responsável pelo controle e acionamento do sistema de retenção;
- *Convenience Control Unit* (CCU), monitora diversas funções, como o sistema antifurto;
- *Transmission Control Unit* (TCU), responsável pelo controle da transmissão do veículo.

O desenvolvimento da ECU foi necessário para o controle eletrônico dos motores, com o objetivo de conter a poluição e melhorar o seu desempenho. Esse sistema de gerenciamento atua no controle da injeção de combustível, controle da mistura, controle da ignição e memória adaptativa. A ECU possui diversos conectores para sensores e atuadores que realizam o envio de informações sobre os mecanismos do veículo para essa central (CAPELLI, 2010).

Para Guimarães (2007), o sistema eletrônico automotivo é semelhante a um computador, pois ele possui características como memória, microprocessador ou microcontrolador e um *software* gravado em sua memória. A conceituação desse sistema se dá por um microcontrolador ou microprocessador, que é o cérebro do dispositivo e responsável por realizar a execução de programas e o gerenciamento de atividades. Ele possui uma memória que internamente executa um *software* embarcado, mais conhecido como *firmware*. Conforme o autor aborda em seu livro, os módulos de gerenciamento eletrônico do veículo realizam a comunicação e o envio de informações através da rede CAN.

A Figura 2 ilustra o visual externo de uma central de gerenciamento de um veículo, elaborada a partir da obra de Capelli (2010).

Figura 2 – ECU



Fonte: Adaptado pelo autor com base em Capelli (2010).

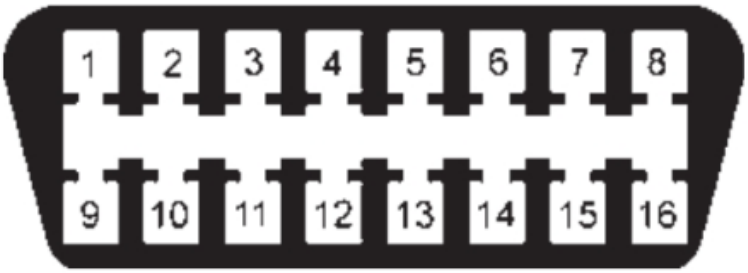
Conforme o autor aborda em seu livro, os módulos possuem conexões para os sensores, que são plugados a essas centrais e transmitem esses dados por meio da rede CAN até a porta de comunicação OBDII.

2.1.5 On-Board Diagnostic II (OBDII)

Segundo Lightner (2005), todos os veículos fabricados a partir de 1995 possuem o sistema de diagnóstico OBD, e o mesmo passou por atualizações durante essa década, tornando-se o sistema OBDII. Esse sistema possui o objetivo de disponibilizar dados referentes ao veículo para leitura, que são realizadas através de um *scanner* de diagnóstico, esse sistema foi desenvolvido principalmente para o controle de emissão de poluentes. Conforme o autor descreve em seu artigo, o monitoramento realizado por esse sistema disponibiliza para leitura dados como velocidade do veículo, rotações por minuto, temperatura do líquido de arrefecimento etc. Todos esses registros são originados da ECU do veículo.

A Figura 3 ilustra o conector OBDII localizado em veículos automotores, conector com 16 pinos, elaborada a partir da obra de Guimarães (2007).

Figura 3 – Conector OBDII 16 pinos



Fonte: Adaptado pelo autor com base em Guimarães (2007).

Segundo Guimarães (2007), o diagnóstico do sistema ocorre a partir dos *Diagnostic Trouble Codes* (DTC), que são códigos pré-programados pelo fabricante e registrados na memória dos módulos eletrônicos, com isso é possível a identificação do que não está funcionando corretamente. Qualquer falha do sistema veicular realiza o registro do DTC em uma memória dedicada a falhas, denominada de memória de falhas.

2.2 Protocolos de comunicação veicular

Os sistemas embarcados possuem protocolos de comunicação que são essenciais para a transmissão e comunicação de dados entre os módulos eletrônicos. Sem a aplicação desses protocolos, não seria possível a integração dos sistemas eletrônicos. Eles possuem características exclusivas e distintas, por meio de transmissão e recepção que ocorrem a comunicação entre os módulos eletrônicos de um veículo (GUIMARÃES, 2007). Os diversos protocolos utilizados em veículos são divididos em grupos e padronizados por entidades como a *Society of Automobile Engineers* (SAE).

2.2.1 Protocolo SAE J1850 Pulse Width Modulation (PWM)

Segundo Oliver (2012), o protocolo SAE J1850 PWM possui um baixo custo de implementação e desenvolvimento, e sua aplicabilidade é dada em modo *on-board* e em modo *off-board* no veículo. Esse protocolo é utilizado pela fabricante de veículos Ford Motor Company, e possui uma taxa de transferência de 41.6 kbps. Fisicamente, ele utiliza os pinos 2 e 10 do conector OBDII, onde o pino 2 é o sinal positivo e o pino 10 o sinal negativo. O tamanho do *frame* de mensagem desse protocolo é de 12 *bytes* com *Cyclic Redundancy Check* (CRC) embutido.

2.2.2 Protocolo SAE J1850 Variable Pulse Width (VPW)

Conforme Oliver (2012), além do protocolo SAE J1850 possuir a versão PWM, existe uma outra versão, denominada de SAE J1850 VPW. Esse protocolo opera com um fio único para a realização da comunicação, e é utilizado pela fabricante de veículos General Motors. A taxa de transferência empregada por esse protocolo é de 10.4 kbps e utiliza apenas o pino 2 do conector para emissão de sinal, e seu *frame* de mensagem possui um tamanho de 12 *bytes* com CRC embutido. Segundo o autor, o barramento de comunicação desse protocolo possui um comprimento máximo de 35 metros.

2.2.3 Controller Area Network (CAN)

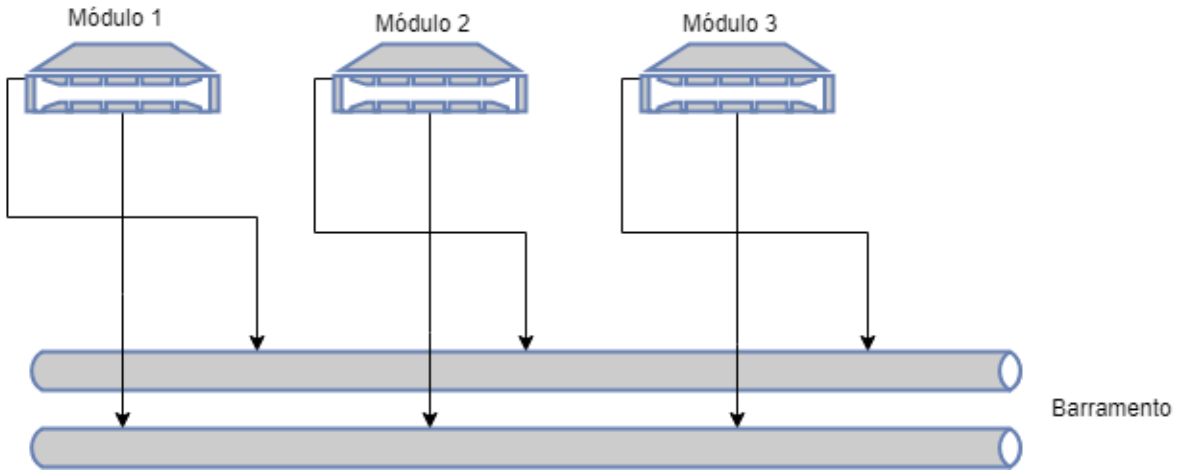
Segundo Capelli (2010), a rede CAN foi desenvolvida em 1986 por Robert Bosch, devido à demanda crescente de componentes eletrônicos embarcados em veículos, e a partir dessa rede foi possível reduzir o custo e o circuito elétrico veicular por meio da centralização do sistema.

Conforme Marques (2004), a troca de informações entre os módulos ocorre de forma modular e a comunicação dessa rede ocorre de forma serial. Os módulos veiculares recebem e transmitem esses dados entre si.

Essa rede trabalha em formato serial síncrono, que torna possível a comunicação entre os módulos de forma que as mensagens enviadas ao barramento ocorrem em intervalos de tempo regulares. O protocolo opera em modo multimestre, onde em determinado período os módulos se tornam mestres e em outro momento se tornam escravos. O envio de mensagens ocorre de forma *multicast*, ou seja, o envio de qualquer informação é realizado para todos os módulos da rede e ele verifica o nível de prioridade dessas mensagens enviadas (GUIMARÃES, 2007).

A Figura 4 exemplifica a rede CAN de um veículo, exibindo o seu barramento e os módulos conectados.

Figura 4 – Rede CAN veicular



Fonte: Do autor (2019).

De acordo com a *International Organization of Standardization* (ISO), a taxa de transmissão é de 1 Mbps (velocidade em um barramento de 40 metros) e suas cargas podem ser de 8 *bytes*.

2.3 Tecnologias de diagnóstico

Uma tecnologia de diagnóstico pode ser definida como um sistema para manutenção onde o objetivo é aumentar e melhorar a qualidade de um produto ou de um serviço. Na próxima seção, será apresentada uma tecnologia de diagnósticos para veículos.

2.3.1 ELM327

Atualmente os veículos têm que possuir uma entrada para o diagnóstico e essa conexão deve permitir a utilização de dispositivos externos para efetuar análises e diagnósticos. O ELM327 foi desenvolvido para realizar uma comunicação com a porta OBDII e uma interface RS232, e o dispositivo possui a capacidade de detectar automaticamente os protocolos, com baixo consumo de energia (DATASHEET, 2019).

De acordo com o Datasheet (2019), a comunicação entre o dispositivo ELM327 e o veículo é realizada por uma rede *ad-hoc*, onde não existe nenhum tipo de autenticação e a comunicação entre a porta de conexão OBDII e o dispositivo é efetuado via serial. A requisição dos dados é realizada por meio de um *software* que solicita ao dispositivo um parâmetro do veículo, como por exemplo as rotações do motor. Essa aquisição de dados ocorre quando o *software* envia um comando hexadecimal, como “0C” para o ELM327, e essa aplicação pode se comunicar com o dispositivo, via *Bluetooth* ou *Wireless Fidelity* (Wi-Fi), variando de acordo com o modelo do *scanner*. Então o ELM327 envia esse valor para ECU e retorna um outro valor hexadecimal, como por exemplo “541B”, onde o primeiro *byte* é denominado de *Diagnostic Test Mode* (DTM) 0x01 que é somado com o valor 0x40, retornando o valor de que tudo ocorreu de forma correta (valor 0x41). Já o segundo *byte* é o *Parameter ID* (PID) solicitado a ECU e o último *byte* corresponde a um valor decimal.

Esses valores são aplicados em uma equação padronizada pela SAE, onde será retornado o valor real das rotações por minuto do motor. Para aplicar esses valores retornados pela ECU, eles devem ser convertidos para decimal e por fim aplicados na Equação 1:

Equação 1 – Equação para cálculo de rotações do motor

$$\mathrm{E} = (256\mathrm{A} + \mathrm{B}) / 4 \qquad (1)$$

Na Equação 1, E é o resultado obtido, a variável A são os km/h, e a variável B o valor recebido da ECU. Existem diversas equações para o cálculo de diferentes sensores, e o exemplo acima é aplicado apenas às rotações por minuto (RPM) do motor.

A Figura 5 exibe um leitor ELM327 *Bluetooth*, *scanner* utilizado para efetuar a leitura do sistema de diagnóstico do veículo por meio do OBDII.

Figura 5 – *Scanner* de leitura ELM327

Fonte: Do autor (2019).

Segundo o Datasheet (2019), esse modelo de *scanner* apresenta recursos somente para realizar leituras de dados da ECU, sem qualquer tipo de tecnologia para realizar escritas na mesma.

2.4 Tecnologia da informação

Segundo Rezende e Abreu (2013), a tecnologia da informação pode ser definida como recursos tecnológicos e computacionais para geração e uso da informação. Os autores fundamentam o conceito de tecnologia da informação com base em componentes de *hardware*, *software*, telecomunicação e gestão. Todos esses recursos unidos devem ser integrados e necessitam da usabilidade de um ser humano, pois sem esse componente fundamental a tecnologia da informação não teria utilidade alguma.

2.4.1 Software

Pressman e Maxim (2016), descrevem que *software* é um elemento lógico e não físico, sendo assim ele não desgasta. Para os autores, o conceito de *software* pode ser explicado como um conjunto de instruções executadas em um computador, e esse mesmo conjunto de dados é ordenado de forma estrutural, que possa manipular e entregar informações de forma correta.

2.4.2 Desenvolvimento de software

Segundo Okuyama et al. (2014), o desenvolvimento de *software* tem como objetivo a elaboração de um sistema computacional, onde o desenvolvimento consiste em um programador possuir a capacidade lógica e matemática para implementação de algoritmos que permitirão a construção de um *software* eficaz. Conforme os autores, esses *softwares* devem passar primeiramente pela área de análise de sistemas, onde serão desenvolvidas as regras principais para serem implementadas nesse programa.

Para Miletto e Bertagnolli (2014), o procedimento inicial para o desenvolvimento de um *software* consiste em adotar um processo de desenvolvimento ou não, mas com sua adoção torna-se possível uma boa documentação, procedimento e distribuição de tarefas.

A Figura 6 retrata o fluxo de desenvolvimento de um *software*, exibindo algumas das principais fases do processo.

Figura 6 – *Workflow* para desenvolvimento de *softwares*

Fonte: Do autor (2019).

Esse processo consiste em diversas atividades que iniciam com a modelagem do sistema, o seu desenvolvimento, o procedimento de teste do *software*, sua implantação e por fim o mantimento ou suporte do sistema.

2.4.3 Aplicativos mobile

Uma aplicação *mobile* pode ser definida como um *software* desenvolvido para plataformas eletrônicas móveis, como *smartphones*, *tablets* etc. Atualmente dois sistemas operacionais *mobile* dominam o mercado, o Android da empresa Google e o iOS da Apple. Esses sistemas possuem instalados diversos aplicativos desenvolvidos com os recursos disponibilizados por essas empresas.

Segundo Deitel (2015), os aplicativos desenvolvidos para plataforma Android são criados com a linguagem de programação Java e o paradigma utilizado no desenvolvimento é a orientação a objeto. Para o desenvolvimento dessas aplicações é necessária uma suíte de desenvolvimento, como o *Android Studio* ou o *Eclipse* com o *plugin* do *Android Software Development Kit* (SDK), conhecimento em programação orientada a objeto (utilizado em linguagens como C#, *Objective-C* ou

C++), caso o programador não possua experiência em Java, onde levará menos tempo para dominar o desenvolvimento para essa plataforma. O autor aborda em seu livro que o aplicativo em etapa de desenvolvimento consiste em componentes gráficos que são criados a partir de arquivos *Extensible Markup Language* (XML) e classes em Java, que são arquivos que possuem toda a lógica de programação necessária para o funcionamento da aplicação.

Com os recursos de classes, componentes e bibliotecas prontas, o desenvolvimento torna-se produtivo e rápido, tornando essas aplicações poderosas com capacidade de tornar os dispositivos móveis com maiores funcionalidades (DEITEL, 2015).

2.4.4 Android

O sistema operacional Android é utilizado em dispositivos móveis. Ele possui código aberto, ou seja, ocorre uma contribuição nesse sistema operacional para o seu desenvolvimento e aprimoramento, e essa iniciativa é liderada e controlada pelo Google (GOOGLE INC, 2019).

Conforme o site do sistema operacional Android, ele foi desenvolvido com base no *kernel* Linux. Sua implementação está desde *smartphones* até satélites desenvolvidos pela NASA, além de ser o sistema operacional mais utilizado do mundo e possuir milhões de aplicativos em sua loja. O Android é projetado para funcionar em qualquer plataforma, e seu desenvolvimento consiste em uma estrutura de camadas que são compostas por uma implementação de *hardware* até aplicações de alto nível.

A Figura 7 demonstra todas as camadas que compõem o sistema operacional *mobile* Android, baseado na documentação oficial do sistema operacional, disponibilizado pelo Google Inc.

Figura 7 – Arquitetura do sistema operacional Android

Fonte: Do autor (2019).

O sistema operacional possui diversos recursos para implementação de aplicativos mobiles, facilitando o desenvolvimento sem a necessidade de criar bibliotecas.

2.4.5 Servidores Web

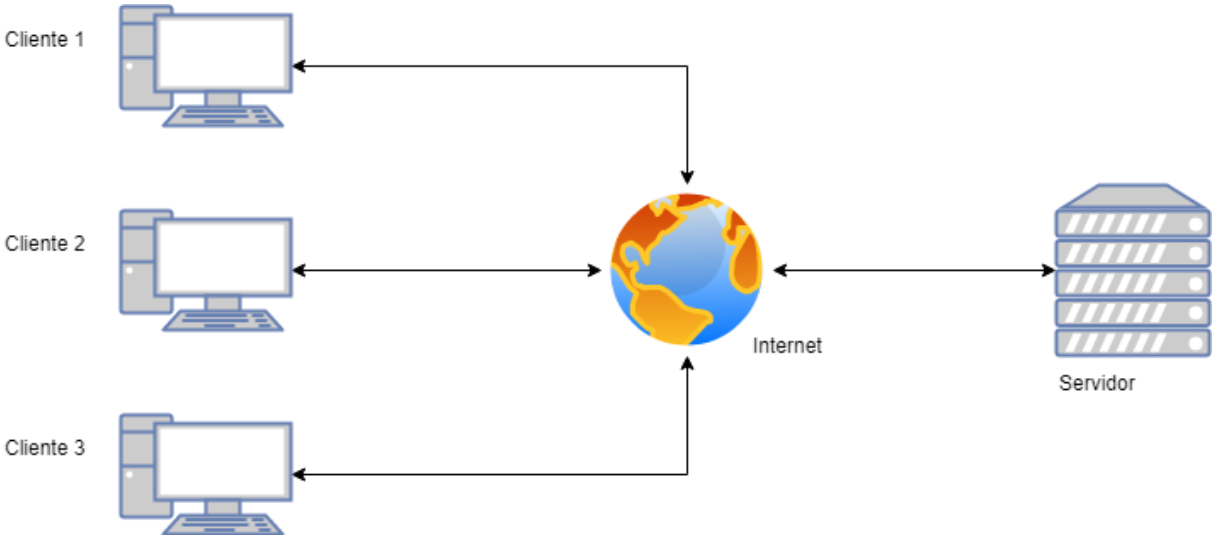
Segundo Didoné e Chaulet (2016), os servidores *web* ou servidores *Hypertext Transfer Protocol* (HTTP) possuem a responsabilidade pelo processamento de requisições HTTP, onde eles realizam a solicitação de páginas de *Internet* entre o cliente e o servidor *web*. O servidor *web* é o recurso que disponibiliza os serviços de cada site na *Internet*, e um exemplo citado pelo autor é de quando um usuário acessa um navegador *web* e digita o endereço de um *site.* Ao pressionar *enter*, o navegador envia uma requisição HTTP para esse *website*. Quando o servidor recebe essa requisição, ele responde com uma página *web*.

Um servidor *web* não pode ser implementado de forma avulsa, devendo conter um conjunto de recursos como servidor de banco de dados, uma linguagem de programação, um sistema operacional e o próprio servidor. Sem esses recursos é praticamente impossível que as páginas *web* tenham um comportamento dinâmico, com apenas o servidor *web* implementado as páginas seriam estáticas e sem nenhum tipo de implementação eficaz, como por exemplo o registro de informações em uma base de dados (DIDONÉ E CHAULET, 2016).

Para Didoné e Chaulet (2016), os servidores *web* possuem a responsabilidade de armazenar páginas *web* e será através desse servidor que ocorrerá a disponibilidade de *websites* que os usuários acessam.

A Figura 8 demonstra o processo de comunicação entre o cliente e um servidor *web*.

Figura 8 – Clientes e servidor se comunicando por meio da *Internet*



Fonte: Do autor (2019).

Além de páginas *web*, esses servidores armazenam um sistema de banco de dados, que é responsável por armazenar as informações que são usadas pelo *website*.

2.5 Comunicação de dados

De acordo com as seções anteriores, o sistema veicular funciona em uma rede CAN que realiza a comunicação entre os módulos veiculares, e os dados transmitidos nessa rede podem ser enviados por meio da comunicação de um aplicativo *mobile* pareado com um *scanner* plugado na porta de entrada da ECU, para um servidor de banco de dados. Entretanto, para que essa comunicação ocorra, é necessário explicar o processo de comunicação de dados.

Segundo Souza (2009) a comunicação entre computadores e sistemas operacionais ocorre por meio da arquitetura *Transmission Control Protocol/Internet*

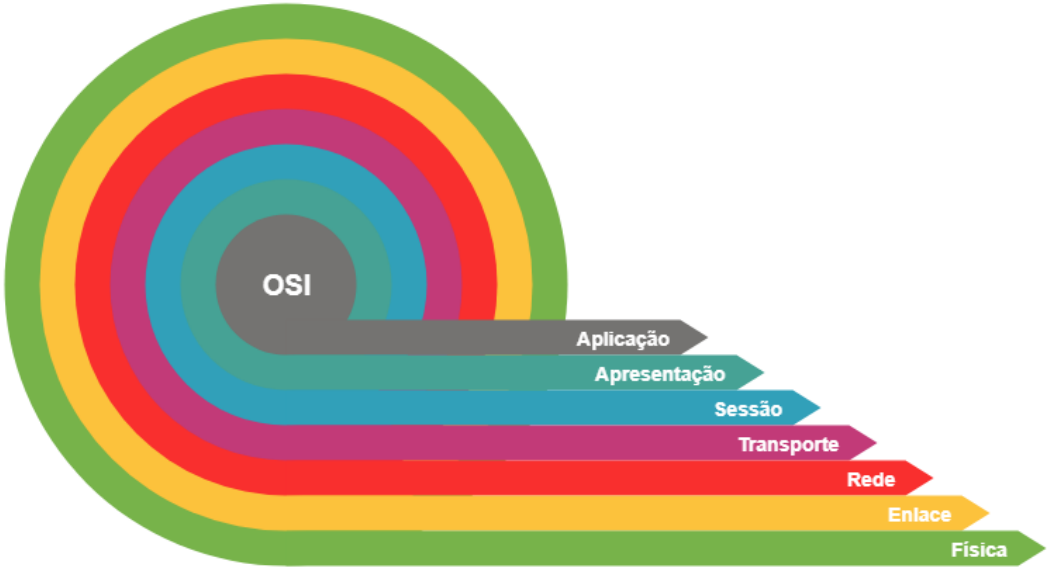
Protocol (TCP/IP), definido como um conjunto de protocolo de comunicação de dados. O autor aborda em seu livro que a conexão de diversas redes é realizada por meio de equipamentos como roteadores que determinam as rotas de comunicação.

2.5.1 TCP/IP

Conforme Schimitt, Peres e Loureiro (2014), o protocolo TCP/IP trabalha com quatro camadas do modelo *Open System Interconnection* (OSI). O modelo OSI é um padrão que os protocolos de rede devem seguir, e esse modelo possui sete camadas, onde cada uma delas é responsável por parte do funcionamento de uma rede.

A Figura 9 foi elaborada baseada na obra de Schmitt, Peres e Loureiro (2014) representando as sete camadas do modelo OSI.

Figura 9 – Sete camadas do modelo OSI



Fonte: Do autor (2019).

O modelo TCP possui as camadas Físicas e de Enlace sobre uma interface de *hardware*, ou seja, uma placa de rede, onde todas essas operações são padronizadas pelo *Institute of Electrical and Electronics Engineers* (IEEE). As demais camadas são implementadas via *software*. Conforme os autores abordam em seu livro, o funcionamento do protocolo TCP/IP trabalham com o conceito de *headers*, onde cada camada contém um espaço de dados denominado de *Protocol Data Unit*

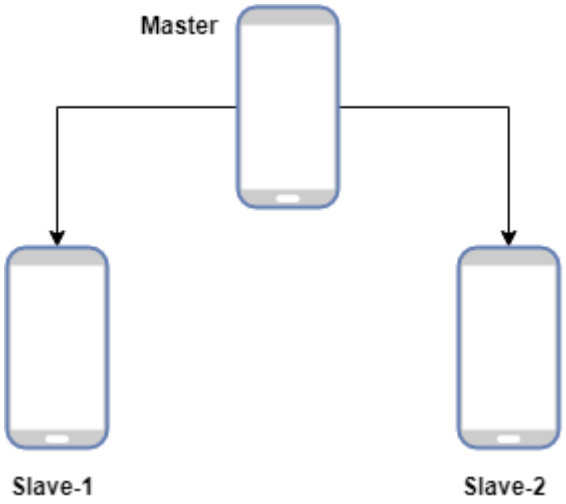
(PDU) e um cabeçalho. Sendo assim, essa camada determina a forma que os dados serão trocados entre aplicações em uma rede, ou seja, essa camada cria um canal lógico de comunicação entre uma aplicação e um servidor. Esse protocolo trabalha com um conceito denominado de serviço orientado a pedido/resposta, cujos dados são enviados de um servidor para um cliente e assim ocorrendo a necessidade de o cliente confirmar se os dados foram recebidos.

2.5.2 Bluetooth

Segundo Lacerra (2008), a tecnologia *Bluetooth* foi lançada em 1998 possuindo alguns problemas em sua implementação, mas após algumas melhorias na comunicação, frequência e implementação, foi lançada uma versão estável no ano de 2004. A versão compreendia uma banda de 2.4 GHz para comunicação entre dispositivos e 79 canais para uso. O funcionamento, segundo o autor, ocorre quando um dispositivo entra na área de alcance e é criado uma rede dinamicamente, onde um dispositivo se torna o *master* e o outro o *slave*, formando uma rede *piconet*.

A Figura 10 apresenta a estruturação de uma rede realizada por meio da comunicação *Bluetooth*.

Figura 10 – Rede de comunicação *Bluetooth* entre dispositivos



Fonte: Do autor (2019).

Essa comunicação utiliza como base *Frequency-Hopping Spread Spectrum* (FHSS), que consiste na comunicação por diferentes canais, onde ocorre de forma aleatória entre os 79 canais em modo sincronizado.

2.6 Aplicação web

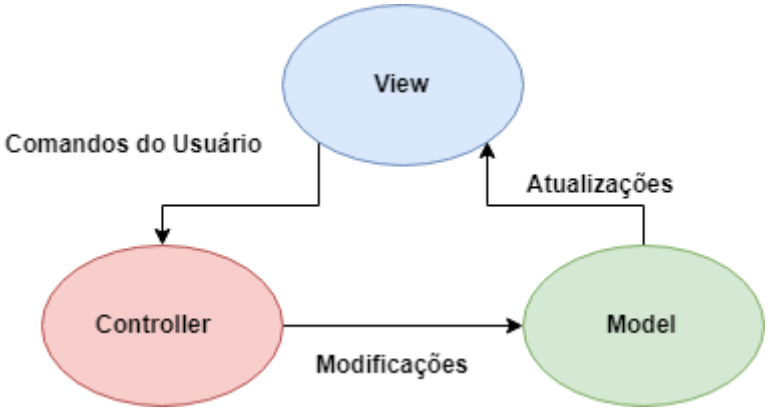
Uma aplicação *web* consiste em sistemas desenvolvidos para serem utilizados por meio de um navegador *web*, onde os mesmos usam tecnologias como *Hypertext Markup Language* (HTML), *JavaScript* e *Cascading Style Sheets* (CSS).

Conforme Machado, Franco e Bertagnolli (2016), o processo de modelagem de um sistema *web* é de extrema importância para o seu desenvolvimento, pois é nessa modelagem que serão definidas as regras de negócio, entidades, modelos e estrutura da base de dados. Esses sistemas são desenvolvidos com base no paradigma orientado a objeto.

Conforme Alves (2015), o *Model View Controller* (MVC) é um padrão de *design* para o desenvolvimento de sistemas, e o seu objetivo é a divisão lógica e de regras de negócio da camada de apresentação de dados.

A Figura 11 representa a arquitetura de um sistema *web* baseado no modelo MVC.

Figura 11 – Arquitetura MVC



Fonte: Do autor (2019).

Segundo Alves (2015), a camada de modelo possui a responsabilidade de manter as regras de negócio, e essa camada realiza a comunicação com o banco de dados. A camada de visão é a interface gráfica exibida para o usuário no navegador *web*, desenvolvida por arquivos HTML, *scripts* em *JavaScript* e estilização por meio de CSS. Por fim, a camada de controle é responsável por realizar a comunicação entre a visão e o modelo, e fica encarregada do controle de ações efetuadas pelo usuário, como por exemplo a realização de um registro através de uma ação do botão de salvar.

2.6.1 HTML e CSS

Segundo Miletto e Bertagnolli (2014), a utilização do HTML é aplicada para o desenvolvimento de páginas *web* e a construção dessas páginas ocorrem pela utilização de *tags* que são interpretadas pelo navegador. Com essas interpretações dos marcadores, formam-se os componentes que são apresentados ao usuário, como formulários, tabelas, botões etc.

O HTML e os padrões para o desenvolvimento *web* são normatizados por uma organização denominada de *World Wide Web* (W3C) (TERUEL, 2014). O HTML pode ser mesclado com outra linguagem, o CSS, responsável por definir regras para formatação de *tags* descritas em arquivos HTML. Com a utilização do CSS é possível alterar a aparência das páginas *web*, como cores, bordas e alinhamentos (MILETTO, BERTAGNOLLI, 2014).

2.6.2 ZK Framework

As informações dessa sessão foram retiradas da página de documentação *online* do *Framework* ZK, fornecido pela desenvolvedora Zkoss.

Conforme a documentação, o ZK é um *framework* de interface gráfica baseado em componentes, possuindo o objetivo de desenvolver aplicações *Rich Internet Application* (RIA) sem a necessidade de conhecimento em *Ajax* ou

JavaScript. O *framework* fornece diversos componentes com vários recursos prontos e a criação desses componentes é realizada através de arquivos XML.

As ações realizadas por um usuário (como o disparo de um evento de clique) são manipuladas por um controlador que realiza a uma determinada atuação para alteração das páginas *web* dos navegadores. O *framework* abrange tratamentos *Ajax*, eliminado a sua implementação e aplicando fortemente o modelo MVC para desenvolvimento de *softwares*.

2.6.3 Java

Segundo Alves (2015), a linguagem de programação Java surge no período em que está começando a *Internet* e essa linguagem é afirmada como adequada para o desenvolvimento *web*.

Conforme Alves (2015), a linguagem de programação Java utiliza o paradigma de programação orientação a objeto e o seu desenvolvimento tem como base a linguagem C++ e *Smalltalk*. Os autores da linguagem utilizaram o C++ para ser a base do Java tornando filha do C++. Java possui uma série de características importantes como multiprocessamento, coletor de lixo da memória para otimização da mesma (*Garbage Collector*) e que é usado para limpar os objetos que não estão sendo utilizados em determinado momento, suportes a interfaces gráficas, programação paralela e distribuída, sendo uma linguagem performática e segura.

Um sistema desenvolvido em Java, para ser executado em um sistema operacional Microsoft Windows, Linux e macOS, deve ser compilado em um primeiro momento, após é executado por uma máquina virtual, denominada de *Java Virtual Machine* (JVM).

Com a JVM, os programas Java se tornam multiplataforma, ou seja, basta possuir o *Java Runtime Environment* (JRE) instalado no sistema operacional para executar qualquer programa Java. Os arquivos onde são escritos os códigos fonte de um programa possuem a extensão .java, e o compilador *javac* gera um *bytecode* que recebe a extensão .*class*, que possui o código de execução. Com essa

tecnologia, não existe a necessidade de recompilar o código para portar a outro sistema operacional (ALVES, 2015).

No próximo capítulo será apresentada a metodologia utilizada para embasamento do trabalho, além da descrição do cenário e da proposta a serem desenvolvidas.

3 METODOLOGIA

No capítulo anterior foram apresentados os principais conceitos, dispositivos e *softwares* necessários para o desenvolvimento do projeto proposto. Neste capítulo é apresentado a metodologia adotada no trabalho, as atividades desenvolvidas o formato de validação dos resultados e a avaliação da proposta.

Segundo Zanella (2013), o estudo de métodos pode ser definido como metodologia, que tem o objetivo de analisar e criticar métodos de investigação. A pesquisa busca o desenvolvimento de novos conhecimentos e tem como finalidade a busca de resoluções para problemas teóricos e práticos.

3.1 Métodos de pesquisa

Conforme os objetivos gerais, essa pesquisa pode ser considerada como exploratória e experimental. Conforme Zanella (2013), essa categoria de pesquisa encaixa-se como uma pesquisa de análise da realidade em busca de conhecimento para posteriormente projetar uma pesquisa descritiva.

Com a finalidade de atingir o objetivo deste trabalho, foi realizada uma análise das tecnologias de diagnóstico embarcadas em um veículo, além da eletrônica convencional do automóvel, por meio da pesquisa de referencial teórico. De acordo com Zanella (2013), essa pesquisa pode ser definida como uma pesquisa bibliográfica, que segundo o autor, possui base em fontes bibliográficas já

publicadas. O referencial teórico tem como objetivo a fundamentação teórica da proposta.

Na sequência, é realizado o desenvolvimento do experimento prático, onde é desenvolvido o aplicativo de comunicação entre a unidade central do veículo e o servidor de banco de dados. Também é desenvolvido um sistema *web*, responsável pelo monitoramento do estado do automóvel que utiliza os dados coletados da central, demonstrando o funcionamento da proposta.

3.2 Cenário

O cenário do experimento é um veículo Volkswagen SpaceFox 2018, onde é realizado a leitura dos dados disponíveis na central de gerenciamento. Esses dados são transmitidos por meio de um dispositivo *scanner* para o aplicativo Android que é responsável em enviar esses dados coletados para um servidor de banco de dados.

Os dados armazenados no banco de dados são analisados por um sistema *web*, responsável por monitorar parâmetros como temperatura do motor, rotações por minuto, nível de combustível, aceleração, velocidade média, quilômetros por hora, temperatura do óleo, geolocalização, entre outras. Esse sistema permite o cadastro de veículos, monitoramento em tempo real, monitoramento geográfico e relatórios sobre o estado do automóvel.

3.3 Aplicações

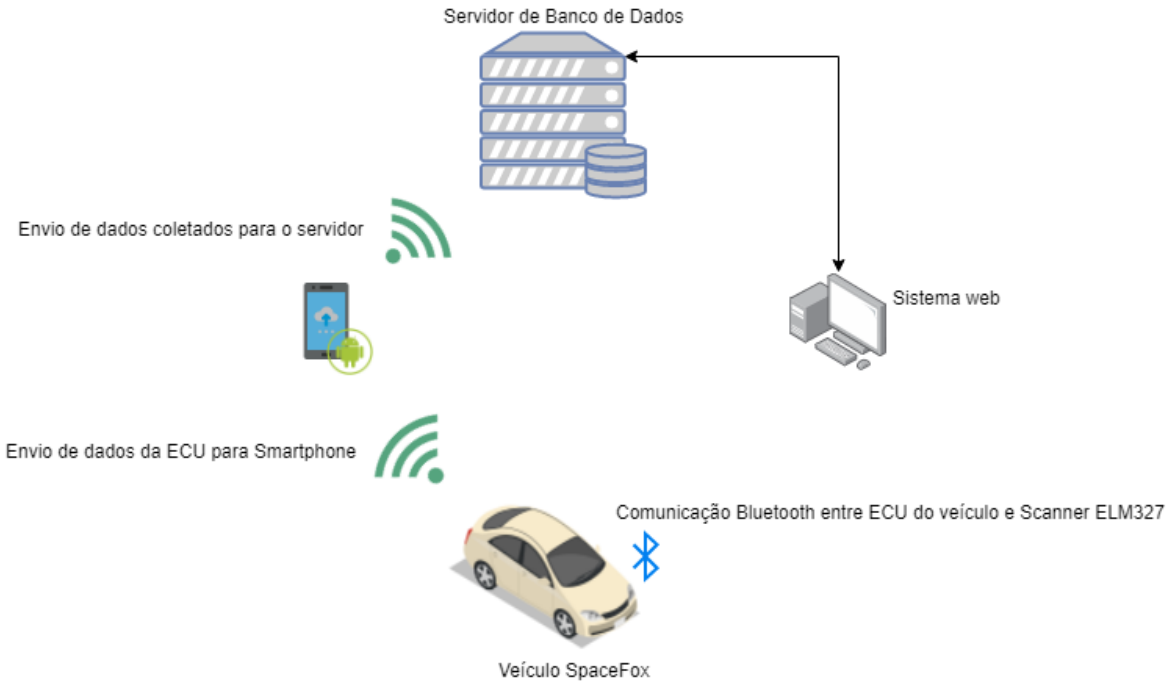
Esse trabalho abrange o desenvolvimento de uma solução para o monitoramento de automóveis, a criação de um aplicativo Android e de um sistema *web*, ambos utilizando linguagem Java e banco de dados. O sistema *web* possui o seu *front-end* desenvolvido com o *framework* ZK.

3.4 Procedimentos técnicos usados na pesquisa

Para avaliação da proposta, são efetuados testes com o veículo e com as aplicações, onde é realizado um percurso com ele para realizar a coleta das informações com o aplicativo desenvolvido. O aplicativo é pareado com o *scanner* plugado ao OBDII e envia os dados ao servidor de banco de dados, por meio de uma conexão com a *Internet*. Com os dados armazenados no servidor de banco de dados, eles são analisados por uma aplicação *web* responsável pelo cadastro de veículos, emissão de relatórios sobre o estado do veículo, monitoramento em tempo real e geolocalização do veículo.

A Figura 12 representa a arquitetura proposta da solução, contendo o veículo, servidor de banco de dados, aplicação *mobile* e sistema *web*.

Figura 12 – Arquitetura de monitoramento de veículos



Fonte: Do autor (2019).

4 DESENVOLVIMENTO

Neste capítulo é apresentado o desenvolvimento do sistema de monitoramento de veículos, abordando o desenvolvimento do protótipo de validação, a modelagem do banco de dados, o desenvolvimento da aplicação *mobile*, da aplicação *web*, do *webservice* e os testes com a aplicação prática.

O sistema é desenvolvido sob a plataforma Microsoft Windows 10, no qual é instalado um banco de dados *MySQL* versão 5, a plataforma de desenvolvimento Java 8 e a suíte de desenvolvimento *NetBeans* versão 8.2, para o desenvolvimento da aplicação *web* e *webservice*. Para o desenvolvimento da aplicação *mobile* foi utilizado a plataforma de desenvolvimento *Android Studio* 3.4.1 e o banco de dados foi modelado na aplicação *MySQL Workbench* 8.0.

4.1 Desenvolvimento do protótipo de validação

O primeiro passo foi encontrar uma biblioteca compatível com o *scanner* ELM327, e em meio a pesquisa foi encontrado uma biblioteca para Android denominada de *Obd-Reader*, encontrada no site de repositórios *Maven*.

A Figura 13 apresenta um esboço do protótipo de aplicação *mobile* para testes.

Figura 13 – Protótipo da aplicação *mobile* desenvolvida para testes

Fonte: Do autor (2019).

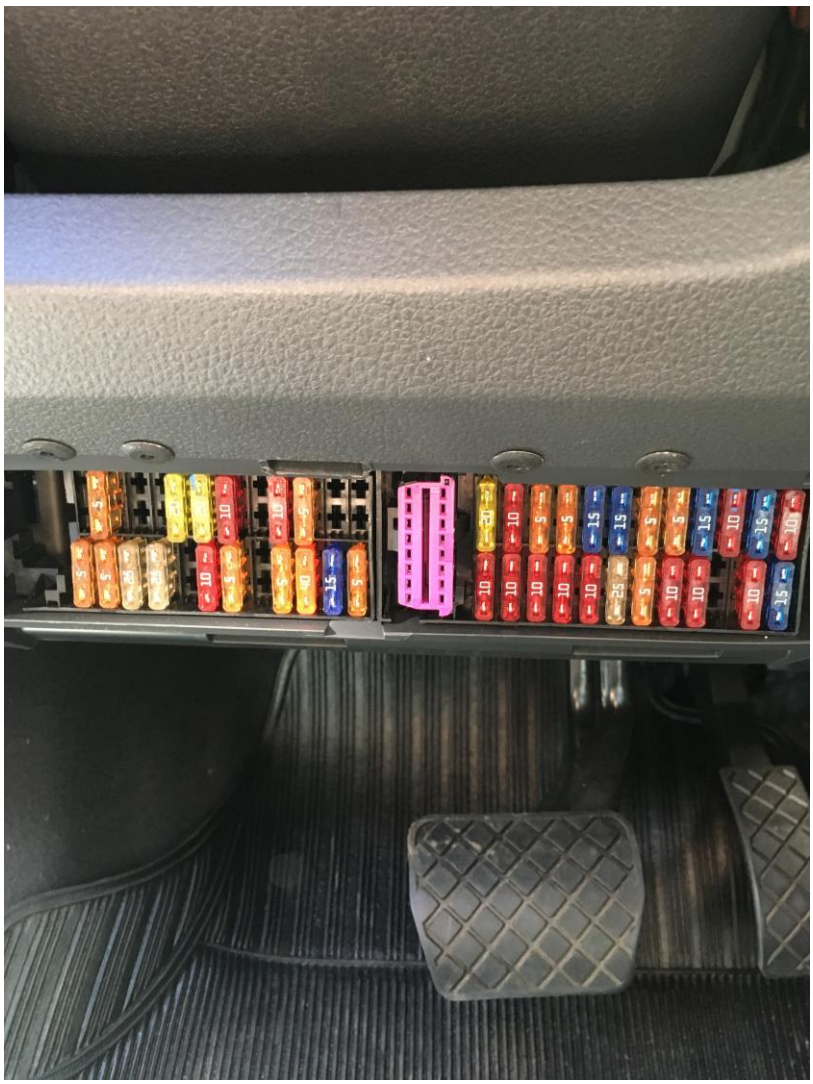
Essa biblioteca envia comandos AT para receber um retorno de dados oriundos da ECU do veículo e o retorno desses dados são valores hexadecimais. Porém, a biblioteca, além de tratar e converter esses dados, realiza o cálculo necessário para apresentar as informações de forma correta, como as rotações, temperatura, pressão barométrica, entre outras, de forma legível ao usuário. Após a configuração da biblioteca, foi realizado o desenvolvimento de uma aplicação *mobile* para fins comprobatórios de funcionamento.

4.1.1 Interface de comunicação veicular

Todos os veículos possuem uma entrada OBDII para leitura de dados da ECU. Essa interface, especificamente no veículo Volkswagen SpaceFox, está localizada junto a caixa de fusíveis abaixo do volante. Nessa entrada é possível plugar um *scanner* ELM327, seja ele de comunicação com fio ou sem fio, pois o conector é padrão.

Na Figura 14 é apresentado o conector OBDII em coloração rosa, do veículo SpaceFox, ano de 2018, para leitura de dados da ECU.

Figura 14 – Entrada OBDII do veículo



Fonte: Do autor (2019).

Por meio dessa porta é possível verificar se o carro apresenta algum problema, qual os erros que possui na central, realizando a leitura da memória de falhas e dependendo do tipo de *scanner* pode ser enviados comandos para central do veículo.

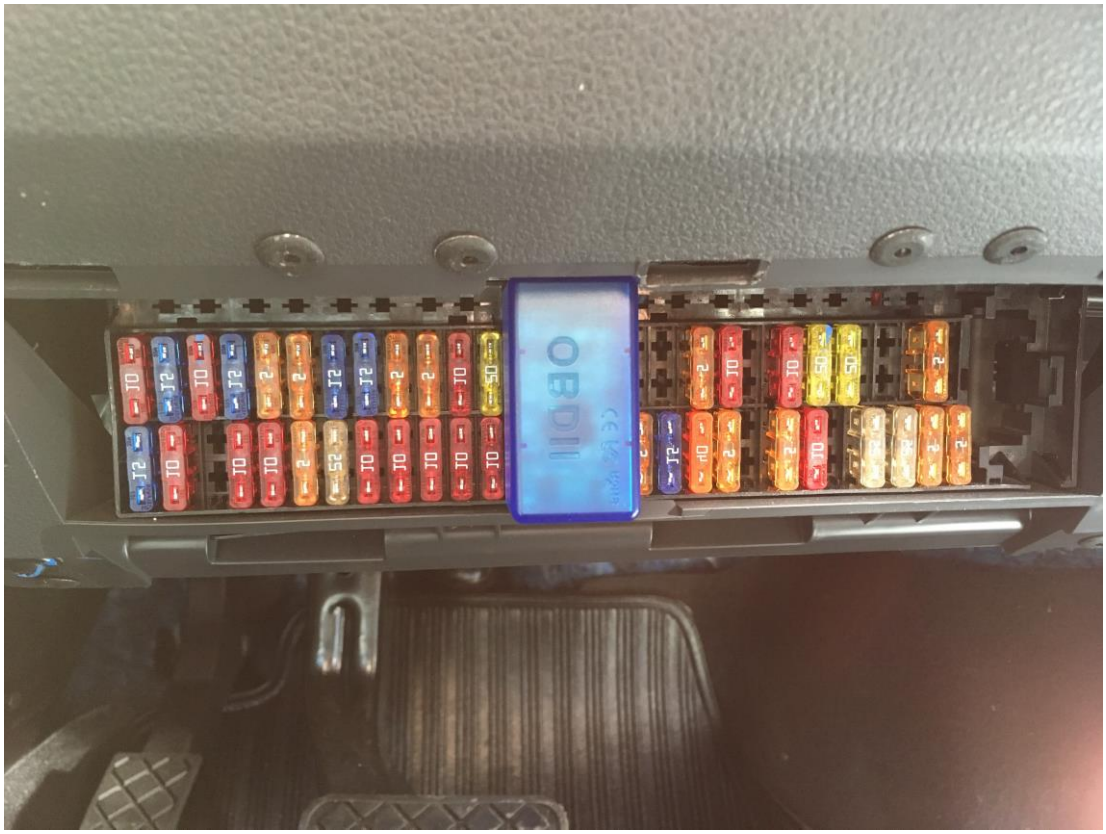
4.1.2 Implementação do scanner ELM327 no veículo

Para realizar as leituras da ECU, foi plugado o *scanner* ELM327, que possui comunicação *Bluetooth* com dispositivos externos e a comunicação com o veículo ocorre de forma física. Ele deve ficar plugado na interface de leitura da ECU.

Esse *scanner* ELM327 efetua o envio de comandos AT no módulo de controle do veículo e retorna um valor.

Na Figura 15 é apresentado o *scanner* ELM327 plugado na porta OBDII do veículo.

Figura 15 – *Scanner* ELM327 plugado na porta OBDII do veículo



Fonte: Do autor (2019).

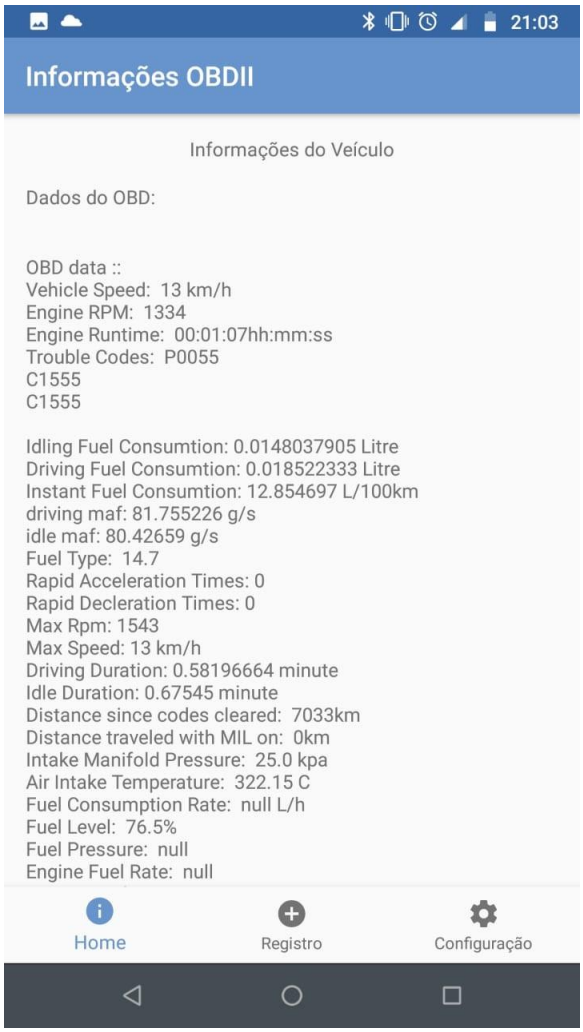
Para os comandos AT serem enviados, deverão possuir um *console* de comunicação ou alguma aplicação, pois o leitor OBDII é apenas um receptor e transmissor, e necessita de um agente para funcionar de forma correta.

4.1.3 Validação do protótipo

Com o desenvolvimento do protótipo, foi possível aprofundar-se no funcionamento da biblioteca e ficou claro onde eram realizadas as chamadas dos objetos populados com os valores carregados da ECU do veículo. Com isso foi possível apresentar uma série de dados enviados em tempo real para a visualização do usuário.

A Figura 16 apresenta a interface do protótipo da aplicação *mobile* com os dados carregados da ECU.

Figura 16 – Protótipo de aplicação *mobile*, com dados da ECU em tempo real



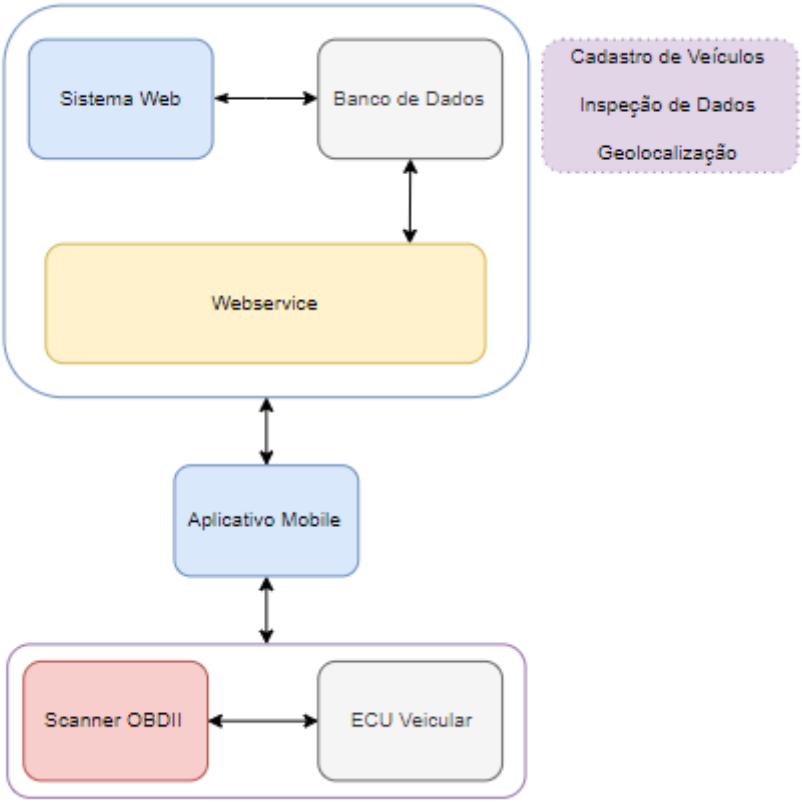
Fonte: Do autor (2019).

O teste com esse protótipo foi realizado com o veículo em movimento, sendo possível validar quais dados seriam úteis. Com essa comprovação de funcionamento, foi possível iniciar o desenvolvimento das aplicações *web* e *mobile*.

4.2 Integração entre os sistemas

A Figura 17, a seguir, apresenta a arquitetura de integração entre a ECU do veículo, o *webservice*, o banco de dados e o sistema *web*, além de mostrar as funções que poderão ser realizadas pelos usuários no sistema *web.*

Figura 17 – Esquema de integração entre os sistemas



Fonte: Do autor (2019).

Nessa arquitetura é possível observar que o aplicativo *mobile* se comunica com a ECU por meio de um *scanner* ELM327, onde o mesmo faz o envio dos dados referentes ao estado atual do veículo por meio de um *webservice* para o banco de dados, no qual o sistema *web* realiza uma conexão para apresentar dados de informação do veículo, geolocalização e informações da viagem realizada ou que está em curso.

O servidor *web*, *webservice* e o banco de dados são instalados em um servidor, onde já está em execução o sistema de produção da empresa, reaproveitando os dispositivos de segurança, como *firewalls*, *proxy*, entre outros.

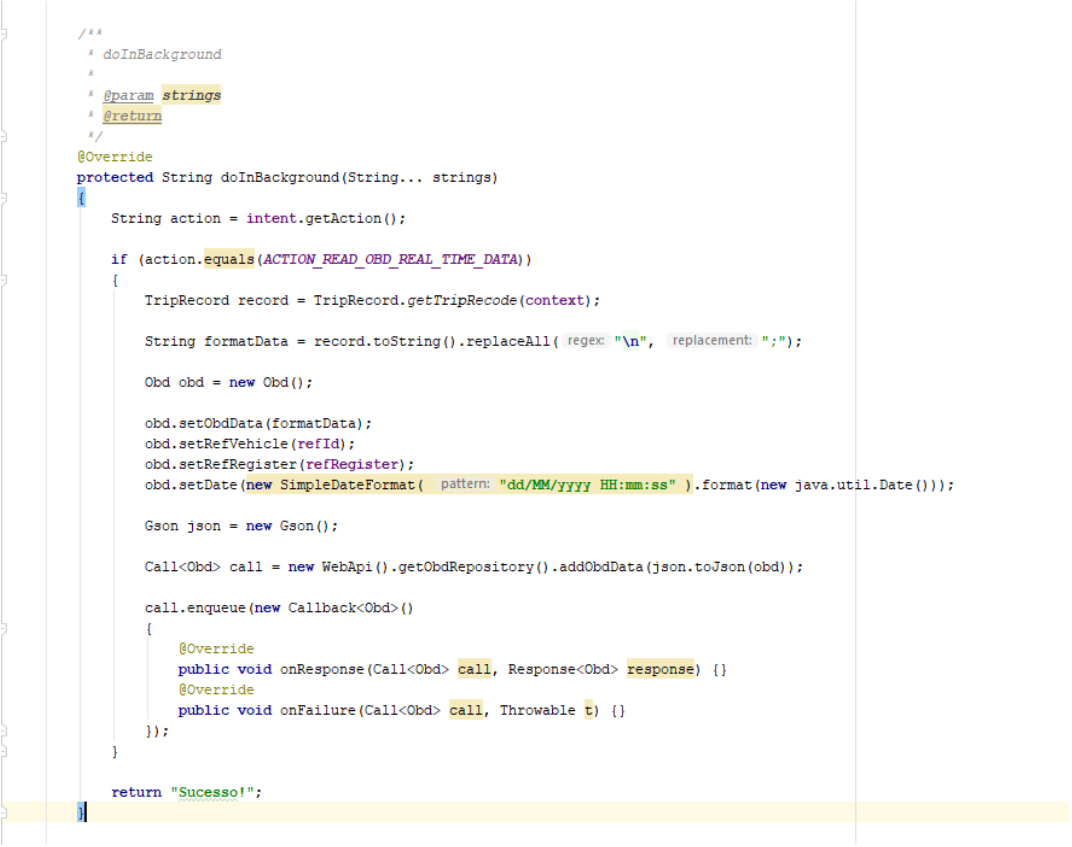
4.3 Desenvolvimento da aplicação mobile

O aplicativo *mobile* foi desenvolvido com a suíte de desenvolvimento *Android Studio*, versão 3.4.1, para sistemas operacionais Android, versão 7 e superiores. Esse aplicativo é responsável por registrar as informações da viagem a ser realizada, comunicar-se com o *scanner* plugado na ECU do veículo e enviar essas informações do estado atual para o banco de dados por meio de um *webservice*, além da geolocalização do dispositivo, para traçar as rotas realizadas pelo veículo.

A biblioteca *Obd-Reader* facilitou a implementação da comunicação e da leitura dos dados originados da ECU do veículo.

A Figura 18 apresenta o método implementado para realizar a leitura dos dados da ECU do veículo.

Figura 18 – Método Java para leitura de dados e envio para o servidor



```
/**
 * doInBackground
 *
 * @param strings
 * @return
 */
@Override
protected String doInBackground(String... strings)
{
    String action = intent.getAction();

    if (action.equals(ACTION_READ_OBD_REAL_TIME_DATA))
    {
        TripRecord record = TripRecord.getTripRecode(context);

        String formatData = record.toString().replaceAll( regex: "\n",  replacement: ";");

        Obd obd = new Obd();

        obd.setObdData(formatData);
        obd.setRefVehicle(refId);
        obd.setRefRegister(refRegister);
        obd.setDate(new SimpleDateFormat(  pattern: "dd/MM/yyyy HH:mm:ss" ).format(new java.util.Date()));

        Gson json = new Gson();

        Call<Obd> call = new WebApi().getObdRepository().addObdData(json.toJson(obd));

        call.enqueue(new Callback<Obd>()
        {
            @Override
            public void onResponse(Call<Obd> call, Response<Obd> response) {}
            @Override
            public void onFailure(Call<Obd> call, Throwable t) {}
        });
    }

    return "Sucesso!";
}
```

Fonte: Do autor (2019).

O envio desses dados para o banco de dados deve ocorrer sem interferência do usuário. Para isso, foram implementados métodos de leitura e o envio desses

dados em determinados períodos, utilizando a classe *TimerTask* do Java. A mesma implementa um *timer*, e a cada minuto envia esses dados para o servidor. No entanto, a leitura dos dados da ECU ocorre em tempo real.

A chamada dos métodos de leitura e o envio de dados ocorre quando a visualização de monitoramento é chamada no aplicativo, iniciando os processos de leitura e de envio de dados para a base de dados, além de apresentar uma interface com um contador de tempo de viagem para o usuário.

4.4 Desenvolvimento do webservice

Com a finalidade de realizar a comunicação entre o aplicativo e o banco de dados, foi desenvolvido um *webservice* em Java para que o aplicativo se comunique e envie as informações do veículo para o banco de dados. O *webservice* realiza especificamente a função de inserir registros da viagem a ser realizada, informações coletadas da ECU do veículo e a geolocalização. A única função de leitura de dados da base de dados aplicada é a de carregar o veículo cadastrado previamente pelo sistema *web*, para efetuar a autenticação na aplicação *mobile*.

Para o desenvolvimento desse *webservice* foi utilizado a biblioteca *Jersey* com arquitetura REST, onde os dados vindos do aplicativo são enviados em formato *JavaScript Object Notation* (JSON), e assim, ocorre a conversão desses objetos para salvar no banco de dados.

4.5 Desenvolvimento do sistema web

O usuário interage com o sistema *web*, cadastrando os veículos previamente e visualizando os dados que são enviados do veículo por meio de inspetores com informações vindas da ECU e um módulo de geolocalização para apresentar as rotas efetuadas.

Uma das tecnologias aplicadas no desenvolvimento do *front-end* foi o *framework* Zkoss, pois essa ferramenta apresenta uma abstração dos componentes

HTML, tornando possível programar igual a aplicações *desktop*, facilitando a implementação e agilizando o processo. Para o módulo de geolocalização, especificamente, foi implementado com uma biblioteca *JavaScript Open Source* denominada de *LeaFlet* que utiliza como motor gráfico para gerar os mapas a plataforma *Open Street Map*.

O *back-end* foi desenvolvido em Java, onde foram realizadas as comunicações com o banco de dados e tratamentos para os dados coletados, devido ao conjunto de informações na base de dados, otimizando assim o seu processamento.

As funcionalidades e os requisitos de *software* foram definidos e serão tratados nas seções seguintes.

4.6 Requisitos e funcionalidades

O aplicativo *mobile* contempla as funcionalidades de registros de viagens, registro das informações da ECU e o registro das coordenadas para formular a sua geolocalização, enquanto que o sistema *web* abrange as funcionalidades de cadastrar os veículos e exibir os dados referente a viagem, os dados da ECU do veículo e a montagem de um mapa com o trajeto realizado.

A aplicação *mobile* apresenta uma tela de login para autenticação de usuários. Essa consulta consome os registros da tabela de *logins* do sistema de gerenciamento da empresa, sem a necessidade de recadastrar os usuários novamente.

A aplicação *mobile* apresenta um seletor de veículos, após o usuário efetuar o *login*. Esse seletor carrega o veículo pré-cadastrado no sistema *web* pelo usuário, o carregamento desse objeto pré-cadastrado é realizado por meio de um ID que está presente no pareamento entre o *smartphone* e o ELM327. Quando o veículo com o ID pré-cadastrado não existir, nenhum veículo é apresentado ao usuário na aplicação *mobile*.

O aplicativo disponibiliza um formulário para registro de viagens a serem realizadas pelo condutor, informando apenas a quilometragem atual e o destino; demais informações, como horário e data, são registradas automaticamente. A leitura de dados da ECU e o envio das coordenadas para geolocalização são processos que funcionam em tempo de execução, paralelamente a outras atividades, executando sem interferência do usuário.

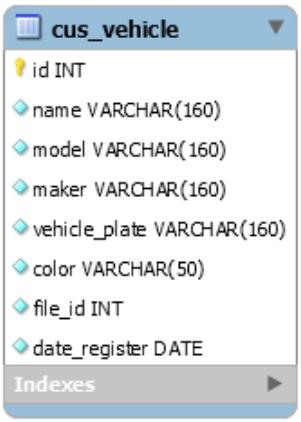
A base de dados contempla quatro tabelas: uma para o registro de veículos, uma para informações da viagem, uma para informações da ECU e uma para sua geolocalização.

4.6.1 Registro de veículos

O sistema *web* apresenta um editor para o cadastro de veículos. Tal cadastro é muito importante para o funcionamento da aplicação no geral, e nele é efetuado um registro do veículo, onde são informadas algumas características como placa, cor, modelo, nome e o seu ID, que é o identificador responsável por diferenciar os veículos e vincular os dados da ECU de forma correta. Esse é o primeiro procedimento que deve ser realizado pelo usuário para o funcionamento da aplicação. Esse registro fica alocado na tabela denominada de CUS_VEHICLES.

A Figura 19 mostra a entidade, que apresenta o modelo da tabela de cadastro de veículos.

Figura 19 – Entidade com modelo do cadastro de veículos



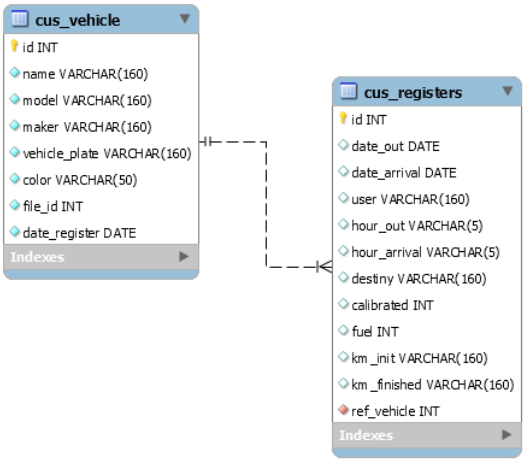
Fonte: Do autor (2019).

4.6.2 Registro de viagens

O registro de viagens é realizado pelo aplicativo *mobile*, quando o condutor realizar uma viagem ele deverá efetuar esse cadastro, essa implementação justifica-se para forçar o usuário utilizar o aplicativo, assim os dados do veículo serão garantidos que serão enviados. Esse registro será vinculado com o veículo previamente cadastrado no sistema web, e fica alocado na tabela denominada de CUS_REGISTERS, esta tabela está vinculada com a tabela de veículos.

A Figura 20 mostra o Diagrama ER-1, que apresenta o modelo da tabela de registros de viagens CUS_REGISTERS.

Figura 20 – Diagrama ER-1 com modelo de registros de viagens



Fonte: Do autor (2019).

4.6.3 Envio de informações do veículo e da geolocalização

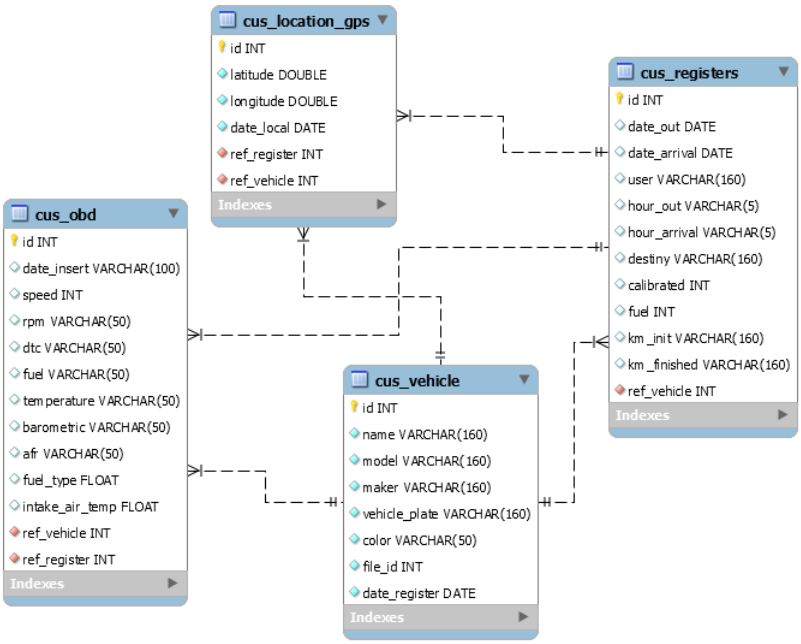
Após o registro da viagem o usuário será redirecionado para uma tela onde será apresentado um contador do período dessa viagem. Nessa tela, são implementados alguns serviços paralelos para a leitura e envio de dados da ECU, e o envio das coordenadas do *smartphone* para montar o mapa de localização do veículo.

Esses dados são lidos em tempo real enviados de 1 em 1 minuto, pois o consumo de dados da rede 3G ou 4G pode ser muito alto e acabar estourando o pacote. Com isso as informações dos dados serão inseridas na base de dados em

conjunto com data, hora e segundos, apresentando precisão para o usuário quando acessar o sistema *web*. As informações do veículo não são apresentadas no aplicativo para o usuário, executando apenas como um processo interno. O registro de informações referente aos dados da ECU é alocado na tabela de CUS_OBD, enquanto os dados referentes a geolocalização são inseridos na tabela CUS_LOCATION_GPS. Ambas as tabelas estão vinculadas com a tabela de registros de viagens e com a tabela de veículos cadastrados.

A Figura 21 mostra o Diagrama ER-2, que apresenta o modelo de tabela do registro de informações da ECU e da geolocalização.

Figura 21 – Diagrama ER-2 com modelo de informações da ECU e geolocalização



Fonte: Do autor (2019).

5 RESULTADOS

Os resultados deste trabalho foram positivos, e o objetivo do projeto foi atingido. O projeto foi desenvolvido, testado e implantado com sucesso.

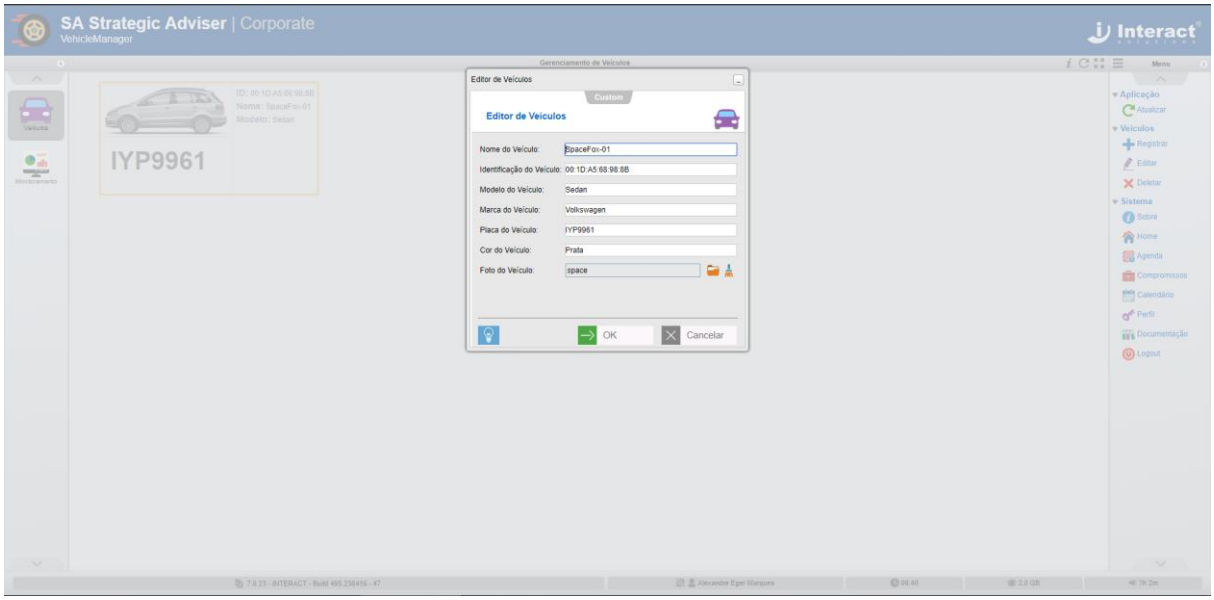
Nesta seção serão apresentadas as funcionalidades do sistema *web*, aplicação *mobile* e testes.

5.1 Cadastro de veículo

De acordo com a regra estipulada para o funcionamento dessa aplicação, o usuário deverá cadastrar primeiramente o veículo no sistema *web* para a aplicação realizar os vínculos com os dados de forma correta. Para isso, o usuário acessa a aplicação *web* e acessa a tela de cadastros, onde é possível cadastrar, editar e excluir um veículo.

O usuário cadastra o veículo por meio de um editor, conforme a Figura 22. Nesse cadastro, algumas informações referentes ao veículo são solicitadas, entre as principais a sua placa e o seu ID. O ID é o endereço *Media Access Control* (MAC) do *scanner* ELM327, e é responsável por vincular os dados do veículo cadastrado com os dados da viagem e da ECU, além de apresentar o automóvel a ser selecionado pelo usuário no momento da viagem.

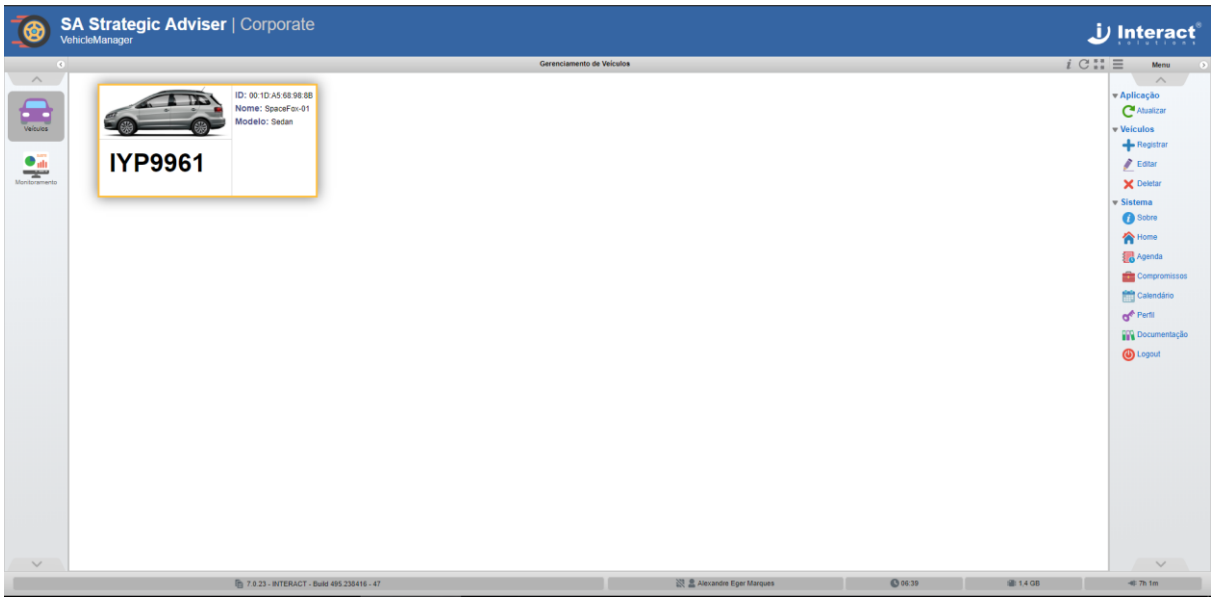
Figura 22 – Editor de registro de veículos



Fonte: Do autor (2019).

Após o veículo ser cadastrado, ele é apresentado ao usuário em um painel de listagem, disponibilizando opções para editar e excluir o registro. A Figura 23 apresenta a listagem do veículo no painel do sistema.

Figura 23 – Veículo cadastrado no sistema



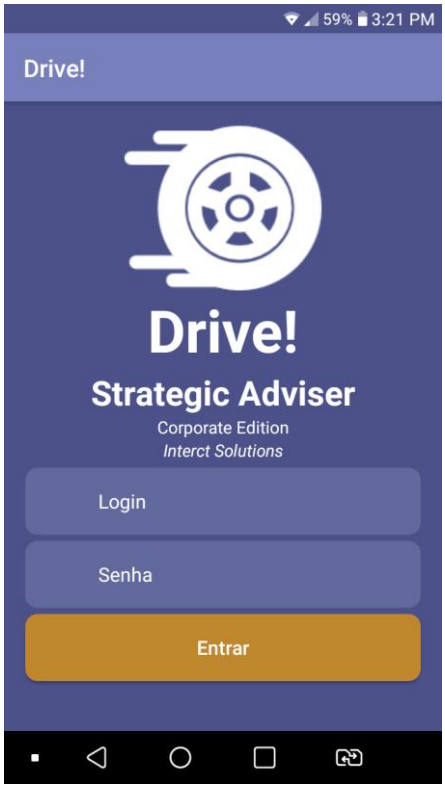
Fonte: Do autor (2019).

O módulo de registros de veículos apresenta uma interface em formato de dashboard, apresentando informações essenciais do veículo cadastrado, sem a necessidade de utilizar inspetores para verificar as informações detalhadas.

5.2 Autenticação no aplicativo mobile

O acesso ao aplicativo *mobile* se dá por meio de uma tela de *login*, conforme a Figura 24, onde o usuário deverá informar o nome e a senha do seu usuário de acesso ao sistema principal da empresa, já que o aplicativo consome a tabela de usuários do sistema, reaproveitando os recursos disponibilizados pelo sistema principal.

Figura 24 – Tela de *login* do aplicativo *mobile*



Fonte: Do autor (2019).

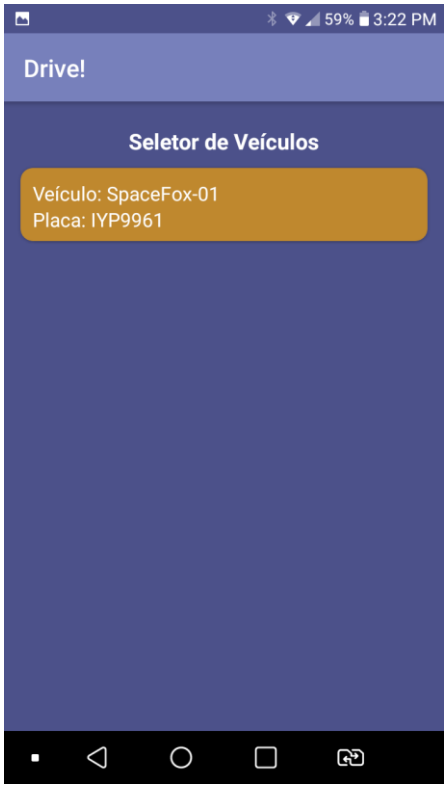
Aproveitando os recursos da plataforma Android, foi aplicado um tratamento para quando o usuário realizar o login e o aplicativo for encerrado, ele manter a sessão logada, solicitando autenticação apenas em caso de atualização ou reinstalação do aplicativo.

5.3 Seletor de veículo cadastrado

Com o *scanner* ELM327 plugado na porta de acesso a ECU do veículo e pareado via *Bluetooth* com o *smartphone*, ao efetuar o *login* no aplicativo, uma consulta é realizada para buscar o veículo cadastrado pelo sistema *web*, por meio do ID do veículo. Utilizando os recursos do Android, é possível buscar o objeto que contém o valor do MAC do *scanner* ELM327. A partir dessa consulta, é exibido ao usuário o veículo cadastrado anteriormente. O usuário poderá selecionar o mesmo e então ele será encaminhado para a tela de registro de viagens.

A Figura 25 apresenta o seletor de veículos, onde um veículo já constava registrado na base de dados.

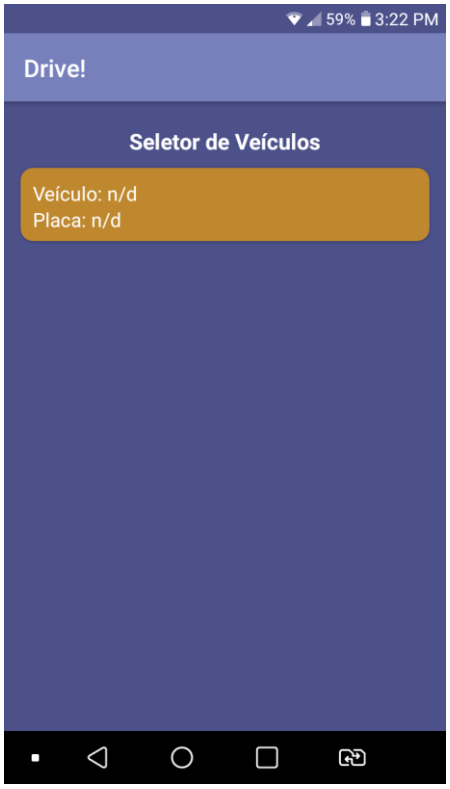
Figura 25 – Tela do seletor de veículos



Fonte: Do autor (2019).

Um tratamento foi adicionado no aplicativo. No caso de o usuário não estar pareado com o dispositivo e efetuar *login* na aplicação, não será possível avançar para os demais processos e o veículo não será apresentado, conforme a Figura 26.

Figura 26 – Tela do seletor de veículos sem registros



Fonte: Do autor (2019).

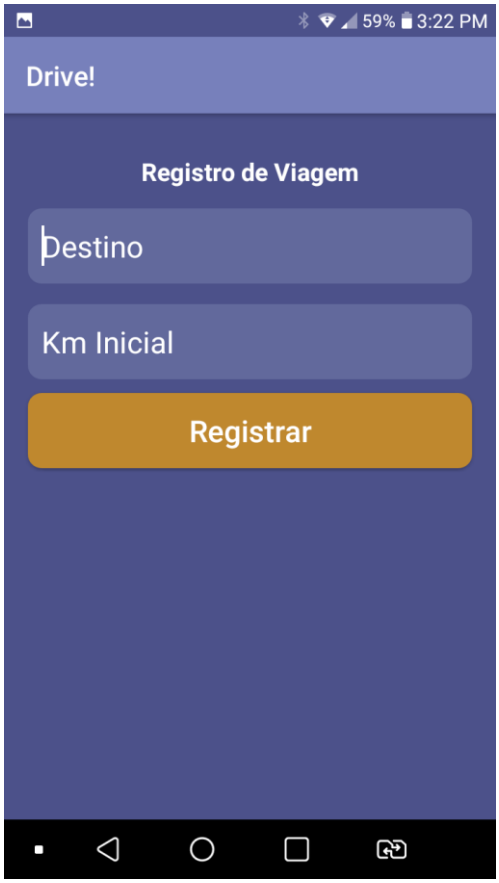
Com esse tratamento o usuário fica impossibilitado de seguir o *workflow* do aplicativo e não ficam brechas abertas, como a possibilidade de efetuar falsos registros de viagem e tentativas de leituras da ECU do veículo.

5.4 Registro de viagens

Nessa etapa, o usuário efetua o registro da viagem a ser realizada no momento, informando apenas a quilometragem atual do veículo e o destino. A quilometragem atual deve ser informada manualmente, pois segundo o *datasheet* do *scanner* ELM327, não é disponibilizado o recurso para efetuar a leitura da quilometragem atual. Demais informações como data e hora são enviadas automaticamente ao servidor quando o usuário efetua o registro.

A Figura 27 apresenta a tela de registro referente a viagem a ser realizada pelo condutor, informando destino e quilometragem inicial.

Figura 27 – Tela do registro de viagens



Fonte: Do autor (2019).

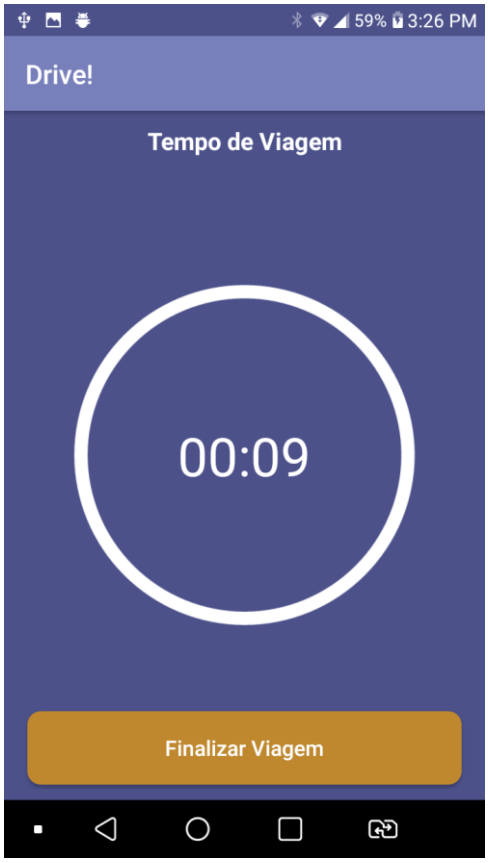
Após o usuário efetuar o registro, ele é redirecionado para tela responsável por enviar os dados do estado do veículo para o servidor, assim como a sua geolocalização.

5.5 Envio de dados para o servidor

Os dados são enviados a cada minuto para o servidor, é efetuado uma leitura dos registros da ECU do veículo em tempo real. Com isso, é possível formular relatórios para os usuários, por meio do sistema *web*. Essas informações não são apresentadas no aplicativo *mobile*, porém é apresentada para o usuário uma tela do tempo de viagem.

A Figura 28 apresenta a tela de processamento de dados e envio para o servidor. Dados da ECU e geolocalização são enviados por meio de alguns processos em tempo de execução.

Figura 28 – Tela de processamento e envio de dados para o servidor



Fonte: Do autor (2019).

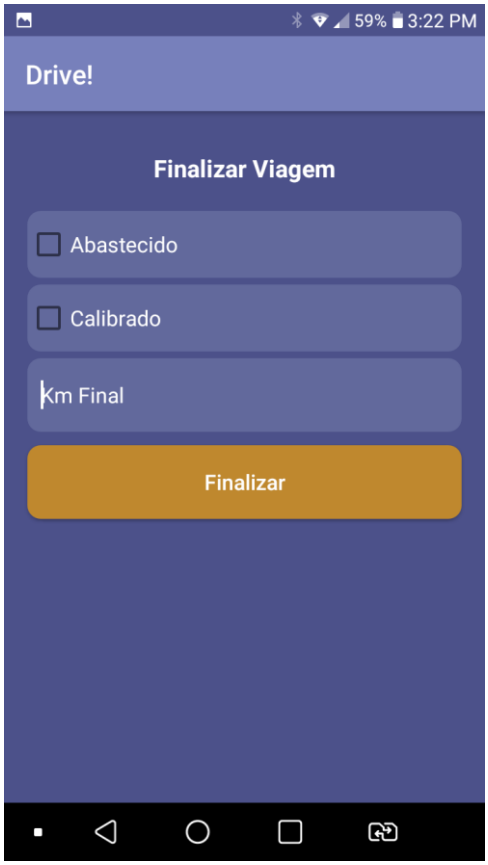
Além das informações da ECU serem enviadas nessa tela, os dados da geolocalização também são enviados, onde a leitura e registro é realizada a cada minuto e por processos distintos funcionando paralelamente.

5.6 Finalização do processo de envio de dados

Para finalizar o envio de dados de uma viagem, o usuário deve pressionar a ação de “Finalizar viagem”, localizada na tela de Tempo de viagem. Nessa etapa, os processos que estão em execução internamente são interrompidos e uma nova tela é apresentada para o usuário preencher algumas informações. Tal ação só ocorre na conclusão de um percurso de uma viagem.

A Figura 29 apresenta a tela de finalização de uma viagem, onde a mesma envia os dados preenchidos pelo usuário e finaliza todos os processos de envio de dados do veículo e geolocalização.

Figura 29 – Tela de finalização de viagem e interrupção do envio de dados



Fonte: Do autor (2019).

Nessa tela, o usuário preenche a quilometragem que o carro apresenta no atual momento. Como já abordado, essa informação não é possível ler com o *scanner* e biblioteca, e se o motorista abasteceu ou calibrou os pneus; se ele marcar, ele executou tal ação.

5.7 Relatório de informações da viagem

Após todo o processo de configuração e registros, foi realizado um percurso com o veículo dentro da cidade de Lajeado, Rio Grande do Sul, para comprovar o funcionamento da aplicação. O percurso traçado compreendia uma pequena quilometragem, e teve uma duração de 15 minutos, suficiente para testar a funcionalidade do sistema.

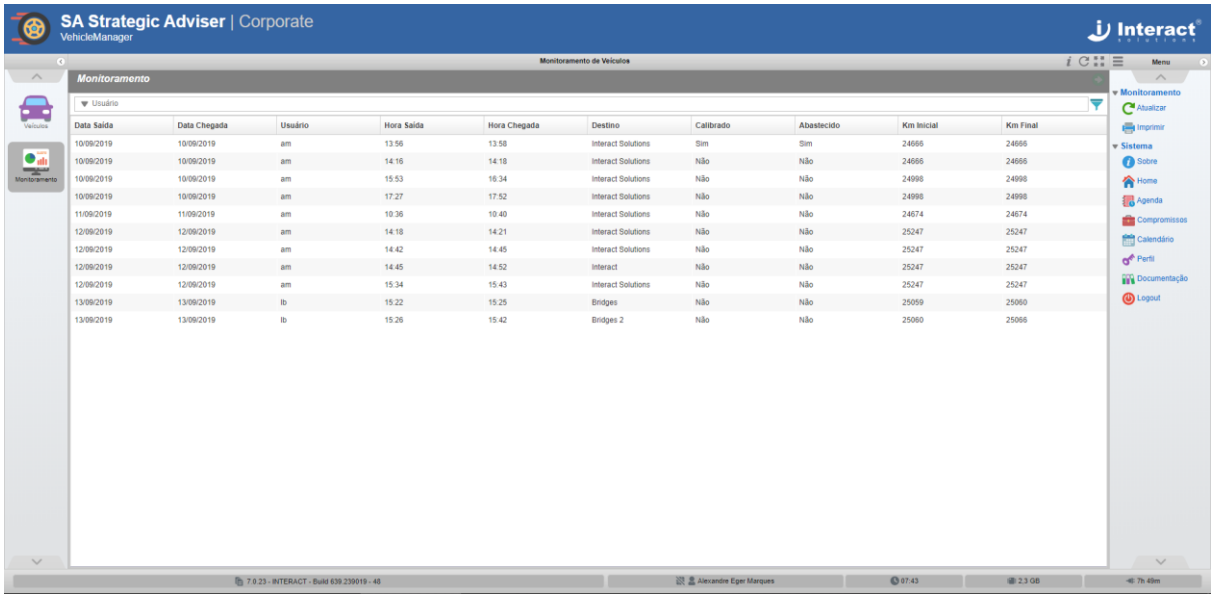
A Figura 30 apresenta a condução do veículo para o envio de informações da ECU e a sua geolocalização.

Figura 30 – Condução do veículo e envio de dados

Fonte: Do autor (2019).

A partir do trajeto realizado, é possível enviar uma série de dados da ECU para o banco de dados e os pontos de geolocalização para formulação do mapa com as rotas realizadas pelo veículo. As viagens realizadas ficam listadas em uma tabela da aplicação *web*, conforme a Figura 31.

Figura 31 – Tabela de listagem das viagens em andamento e realizadas

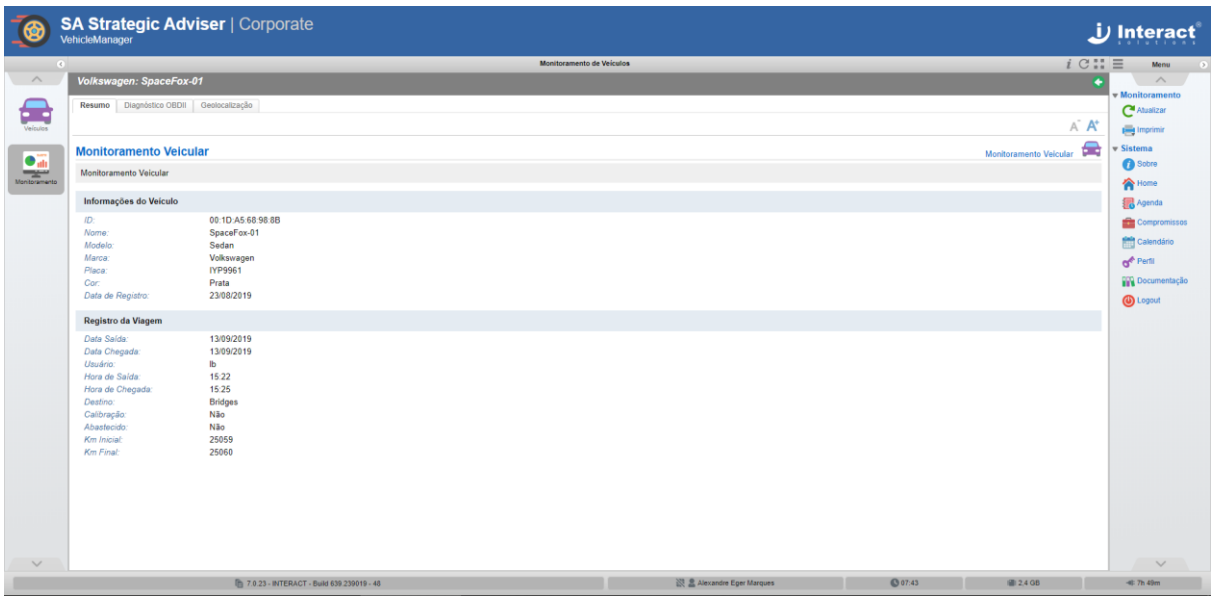


Fonte: Do autor (2019).

É possível selecionar uma viagem e inspecioná-la, onde são carregadas as informações em detalhes referentes à viagem selecionada. São exibidos os dados enviados da ECU do veículo e o trajeto realizado pelo veículo.

A Figura 32 apresenta as informações em detalhes da viagem e habilitando as abas de informações da ECU, no caso Diagnóstico OBDII e a aba de Geolocalização.

Figura 32 – Informações detalhadas da viagem



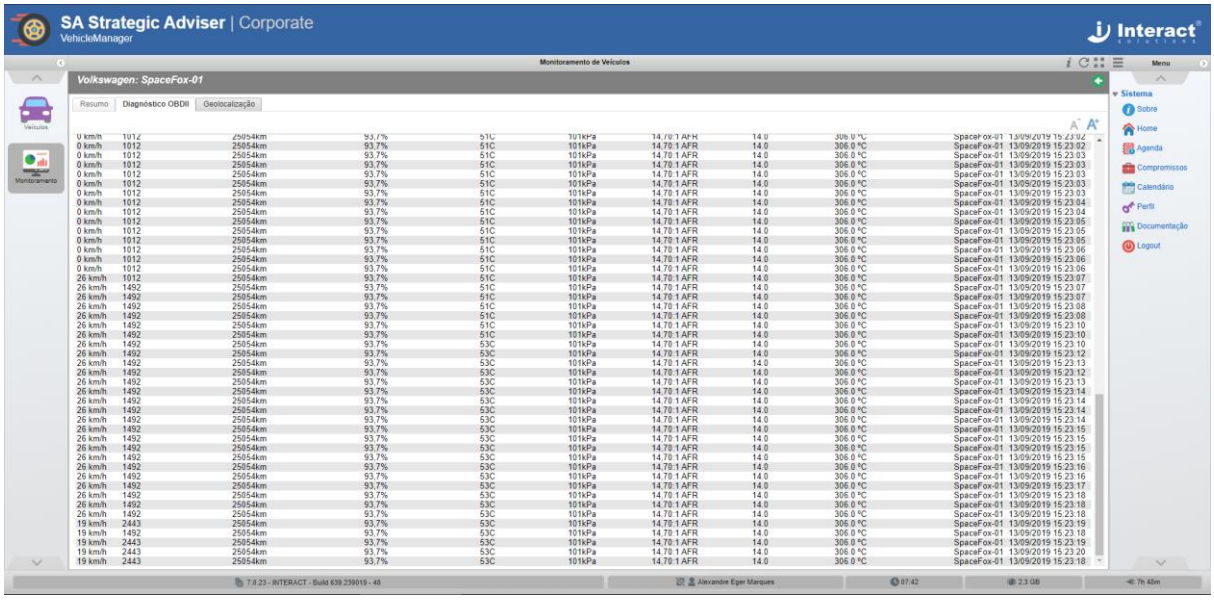
Fonte: Do autor (2019).

5.7.1 Envio de informações do veículo e da geolocalização

Com a conclusão do percurso, diversos registros foram inseridos na base de dados, referentes à ECU do veículo e coordenadas de geolocalização. Nesse trajeto, foi possível avaliar uma média de velocidade que o condutor ficou em determinados períodos, o percentual de combustível no tanque, a pressão barométrica, a temperatura do motor baseada no sensor de líquido de arrefecimento do automóvel, o tipo de combustível, a temperatura na câmara de saída após a combustão e a quilometragem que o veículo se encontrava desde a última vez que os erros da ECU foram apagados.

A Figura 33 apresenta o inspetor de dados da ECU, exibindo os dados coletados. Esses dados variam de segundo em segundo, onde a cada minuto um *array* de informações é enviado, podendo muitas vezes virem repetidos, mas cumprindo o seu papel em informar o estado atual do veículo.

Figura 33 – Relatório de informações da ECU do veículo

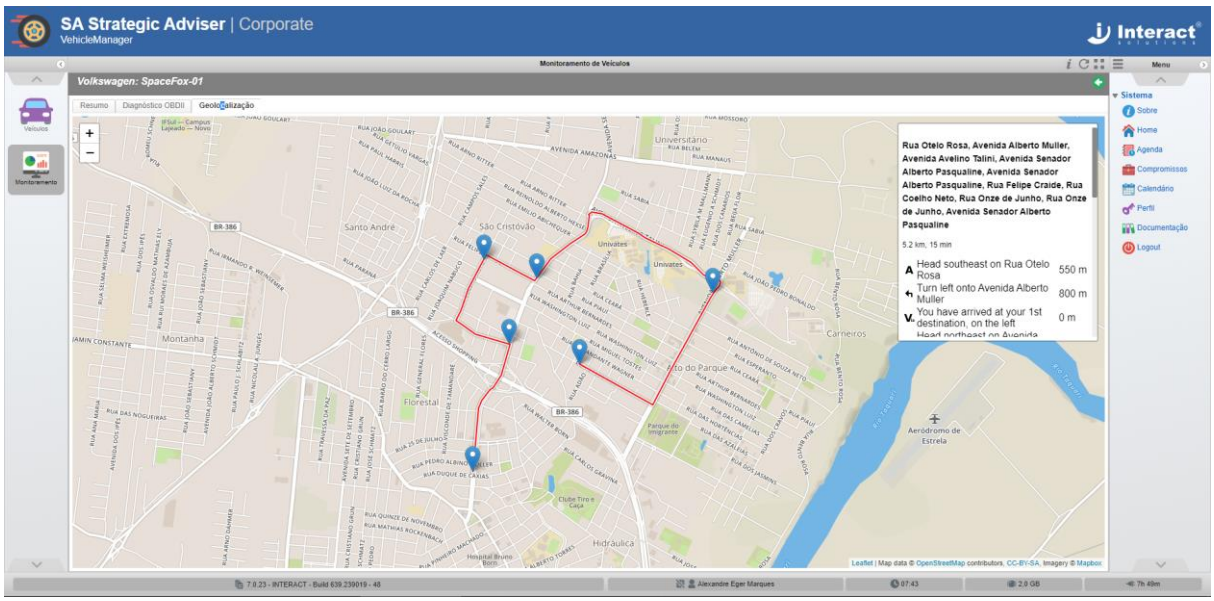


Fonte: Do autor (2019).

O modulo de geolocalização é encontrado na mesma tela que as informações da viagem e da ECU do veículo, e é constituído de apresentar os pontos que o veículo passou por meio de coordenadas e enviá-las ao servidor. Com essas coordenadas, um mapa traçando as rotas é renderizado e exibido ao usuário.

A Figura 34 apresenta o mapa com o percurso e a geolocalização dos pontos que o veículo passou.

Figura 34 – Geolocalização do veículo



Fonte: Do autor (2019).

Em determinados pontos o envio de coordenadas pode falhar, pois a leitura desses dados é realizada do gps do *smartphone*, sem nenhum tipo de atualização originada de servidores da *internet*.

5.8 Validação

O sistema foi instalado na empresa no mês de setembro de 2019. O cadastro de veículos foi realizado após a implantação da base de dados, do *webservice* e da aplicação *web*. Durante os primeiros usos, foram realizados procedimentos de testes de bancada para buscar possíveis falhas, como inconsistência em relatórios e problemas na renderização dos mapas.

Foram realizados ajustes em erros do sistema *web* e do *webservice*, como falhas em cadastros, no retorno de registros que estavam inconsistentes, além de correções de problemas de *cache* para renderizar o mapa de geolocalização do veículo. O aplicativo *mobile* passou por uma revisão, onde foram ajustados pontos de processamento para evitar travamentos e encerramento do aplicativo de forma

inesperada. Para isso, aplicou-se um aumento no *heap* de memória do *buffer* de leitura de dados ao comunicar-se com o *scanner* ELM327, além da divisão dos processos de leitura e envio, pois causavam o travamento da aplicação.

Foram realizados testes durante uma semana na empresa, onde alguns percursos de curta distância foram realizados. O sistema apresentou-se eficaz, pois os dados referentes ao estado do veículo foram enviados com sucesso ao servidor de banco de dados e apresentados de forma legitima por meio do sistema *web*, além da funcionalidade da geolocalização, onde o mapa demonstrou corretamente as rotas traçadas.

O sistema atualmente não está em produção devido a uma solicitação da gerência, que pretende avançar no projeto com novas funcionalidades e integração. Além disso, há uma intenção estratégica de diminuir o consumo de dados da aplicação, pois como existe um enorme envio de dados para o servidor, tal custo estava sendo elevado. Em uma semana foram coletados 4069 registros do veículo, por isso, o processo deverá ser otimizado para diminuir o tráfego de dados e assim integrar com demais funções do sistema da empresa, no futuro, como o processo de manutenção de veículos e demais módulos.

5.9 Dificuldades

As dificuldades mais relevantes encontradas durante o desenvolvimento deste trabalho foram na parte prática e na execução de testes com o envio de dados do veículo para o servidor, pois os testes de comunicação inicial precisavam ser realizados no estacionamento da empresa, e eram impossibilitados nos dias de chuva e quando o veículo não se encontrava na empresa.

Outra dificuldade encontrada foi durante o processo de *debug* da aplicação *mobile*, pois em um primeiro momento era necessário a realização de ajustes no setor de desenvolvimento da empresa, e para validar o funcionamento era preciso a rodagem em um trajeto para simular e coletar os *logs* de problemas. Essa dificuldade foi amenizada com a utilização de um notebook com as aplicações de desenvolvimento para Android instalada, sendo então possível realizar os testes e os *debugs* dentro do veículo.

A última dificuldade reportada refere-se à biblioteca *Obd-Reader*, que apresenta grande problema de despejo de memória. Tal problema foi contornado aumentando o *heap* de memória da aplicação *mobile*.

6 CONCLUSÃO

Este trabalho apresentou o desenvolvimento e implantação de um sistema para monitoramento de veículos, por meio do envio de dados da ECU para um sistema de monitoramento *web*. Esse sistema foi implementado na empresa Interact Solutions da cidade de Lajeado, no Rio Grande do Sul. A proposta do trabalho foi baseada em um referencial teórico e pesquisa de campo, buscando os recursos para o desenvolvimento do projeto.

O sistema *web* facilitou o entendimento, apresentando poucas funcionalidades de interação com o usuário, porém apresenta uma interface rica de monitoramento dos veículos, além de abrir outros caminhos para futuras interações com demais ferramentas existentes no sistema de gestão da empresa.

O aplicativo *mobile* também apresenta uma interface rica e pouca interação com o usuário, já que o seu objetivo é apenas enviar informações da viagem e da situação do veículo para o servidor de banco de dados.

O presente trabalho apresenta muitos pontos positivos, entre eles a automação dessa área da empresa e a abertura de possibilidades de utilização desses dados em outras aplicações, além de informar o estado de integridade do veículo e a sua geolocalização, a partir das coordenadas do gps do *smartphone*, proporcionando segurança ao condutor e evitando altos prejuízos em caso de falha de algum sistema mecânico do veículo.

Pode-se concluir que o sistema de monitoramento veicular é uma solução moderna e eficaz para ajudar a empresa a gerenciar o estado de integridade da frota de veículos, integrando os mesmos ao processo da empresa.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, W. P. **Java para Web**. São Paulo: Editora Érica, 2015. 385 p.

CAPELLI, A. **Eletroeletrônica Automotiva**: Injeção Eletrônica, Arquitetura do Motor e Sistemas Embarcados. São Paulo: Editora Érica, 2010. 369 p.

DEITEL, P., DEITEL, H., DEITEL, A. **Android:** Como Programar. 2. ed. Porto Alegre: Bookman, 2015. 669 p.

DIDONÉ, D., CHAULET, F. J. **Implantação e Administração de Serviços Web**. Recife: IFPE, 2016. 88 p.

ELMELETRONICS. **Datasheet.** Disponível em: <https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf>. Acesso em: 14 mar. 2019.

GOOGLE INC. **Android Source.** Disponível em: <https://source.android.com/setup>. Acesso em: 23 abr. 2019.

GUIMARÃES, A. A. **Eletrônica Embarcada Automotiva.** São Paulo: Editora Érica, 2007. 329 p.

ISO**.** Disponível em: <https://www.iso.org/home.html>. Acesso em: 25 abr. 2019.

LACERRA, I. D. **Bluetooth.** Disponível em: <http://grenoble.ime.usp.br/~gold/cursos/2008/movel/mono/bluetooth.pdf>. Acesso em: 01 abr. 2019.

LAGANÁ, A. A. M. **Apostila de sensores.** Disponível em: <https://edisciplinas.usp.br/pluginfile.php/241350/mod_resource/content/1/Apostila%2BSensores03.pdf>. Acesso em: 06 mar. 2019.

LIGHTNER, B. D. **AVR-Based Fuel Consumption Gauge.** Disponível em: <http://www.lightner.net/lightner/bruce/Lightner-183.pdf>. Acesso em: 04 abr. 2019.

MACHADO, R. P., FRANCO, M. H. I., BERTAGNOLLI, S. C. **Desenvolvimento de Software III:** Programação de Sistemas Web Orientada a Objetos em Java. Porto Alegre: Bookman, 2016. 210 p.

MARQUES, Marco Antônio. **CAN Automotivo Sistema de Monitoramento.** Disponível em: <http://alvarestech.com/temp/murilo/obd-device-can.pdf>. Acesso em: 23 mar. 2019.

MILETTO, E. M.; BERTAGNOLLI, S. D. C. **Desenvolvimento de Software II:** Introdução ao Desenvolvimento Web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014. 276 p.

OKUYAMA, F. Y., MILETTO, E. M., NICOLAO, M. **Desenvolvimento de Software I**. Porto Alegre: Bookman, 2014. 229 p.

OLIVEIRA, A. S; ANDRADE, F. S. **Sistemas Embarcados:** Hardware e Firmware na Prática. 2. ed. São Paulo: Editora Érica, 2010. 321 p.

OLIVER, D. J. **Implementando o protocolo J1850.** Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.506.6682&rep=rep1&type=pdf>. Acesso em: 08 mar. 2019.

PERES, A., LOUREIRO, C. A. H., SCHMITT, M. A. R. **Redes de Computadores II:** Níveis de Transporte e Rede. Porto Alegre: Bookman, 2014. 121 p.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software:** Uma Abordagem Profissional. 8. ed. Porto Alegre: AMGH Editora LTDA, 2016. 940 p.

REZENDE, D. A., ABREU, A. F. **Tecnologia da Informação:** Aplicada a Sistemas de Informação Empresariais. 9. ed. São Paulo: Atlas, 2013. 346 p.

SALOMAN, S. **Sensores e Sistema de Controle na Indústria.** 2. ed. Rio de Janeiro: Livros Técnicos e Científicos Editora Ltda, 2012. 510 p.

SOUZA, L. B. **TCP/IP & Conectividade em Redes.** 5. ed. São Paulo: Editora Érica, 2009. 193 p.

TERUEL, E. C. **HTML 5:** Guia Prático. 2. ed. São Paulo: Editora Érica, 2014. 337 p.

THOMAZINI, D., ALBUQUERQUE, P. U. B. **Sensores Industriais:** Fundamentos e Aplicações. 8. ed. São Paulo: Érica, 2011. 225 p.

ZANELLA, L. C. H. **Metodologia de Pesquisa**. 2. ed. Florianópolis: UFSC, 2013. 134 p.

ZKOSS. **Documentação ZK.** Disponível em: <https://www.zkoss.org/documentation>. Acesso em: 28 abr. 2019.